M-M-Max, the New AMIGA Television Star

Amazing COMPUTING™

Your Original AMIGA™ Monthly Resource

Volume 2 Number 10
US $3.50 Canada $4.50

PROGRAMMING:

AMIGA Pascal Reviewed
AMIGABasic Structures
C Animation
And MORE!

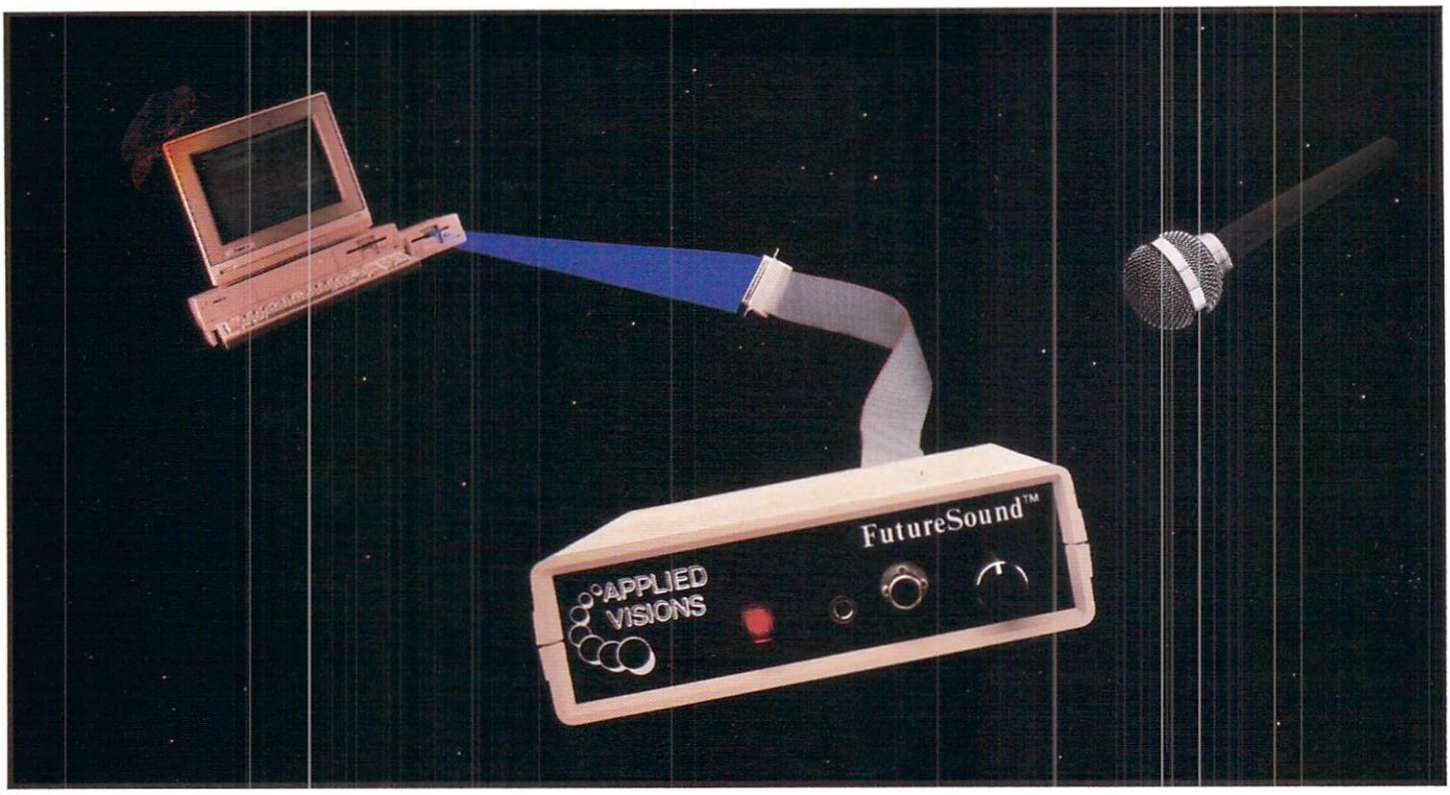
Business Programming:
dBMan

New Amazing Column!
Bug Bytes

Photo Tips:
Taking The Perfect Screen Shot

AMIGA Forum Transcripts:
Software Publishing Conference

```
opencode(direction)

/*   This installs the VU meter.

     If this is to install the output routine:

     Get the Port structure for the Sampler.
     If the Sampler already has its output
     code stolen, (samplerport->show has somebody's
     port id) do nothing.  Else, open the window,
     give samplerport->outcode insertoutcode
     and put the Sampler's output code pigs
     in samplercode.  Set samplerport->show to
     this port's id, so no other module will
     monkey with it.
*/

char direction;

{
  short i;
  struct Port *samplerport;
  if (!direction) {
    samplerport = (struct Port *)
      MidiPort(FindMidiPort("sampler"));
    if (samplerport) {
      if (samplerport->show) {
        CloseMidiPort(samplerport->show,0);
      }
      if (!samplerport->show) {
        vuwindow = (struct Window *)
          OpenWindow(&NewWindowStructure);
        if (vuwindow) {
          totalvolume = 0;
          for (i=0; i < 128; i++)
            velocity[i] = 0;
          sampleroutcode = samplerport->outcode;
          samplerport->outcode = insertoutcode;
          samplerport->show = thisport;
          return(1);
        }
      }
    }
    return(0);
  }

losecode(direction)

     This deactivates the VU meter.

     return the Sampler's output r
     t structure and clear
     ->show.  Close the display
     to
     sampl
     windo

*/
```



"Open the pod bay doors, HAL..."

Programmers cast their vote!

Right now, leading software developers are hard at work on the next generation of Amiga® products. To add the spectacular sound effects we've all come to expect from Amiga software, they are overwhelmingly choosing one sound recording package... FutureSound. As one developer put it, "FutureSound should be standard equipment for the Amiga."

FutureSound the clear winner...

Why has FutureSound become the clear choice for digital sound sampling on the Amiga? The reason is obvious: a hardware design that has left nothing out. FutureSound includes two input sources, each with its own amplifier, one for a microphone and one for direct recording; input volume control; high speed 8-bit parallel interface, complete with an additional printer port; extra filters that take care of everything from background hiss to interference from the monitor; and of course, a microphone so that you can begin recording immediately.

What about software?

FutureSound transforms your Amiga into a powerful, multi-track recording studio. Of course, this innovative software package provides you with all the basic recording features you expect. But with FutureSound, this is just the beginning. A forty-page manual will guide you through such features as variable sampling rates, visual editing, mixing, special effects generation, and more. A major software publisher is soon to release a simulation with an engine roar that will rattle your teeth. This incredible reverberation effect was designed with FutureSound's software.

APPLIED VISIONS

Question: What can a 300 pound space creature do with these sounds?

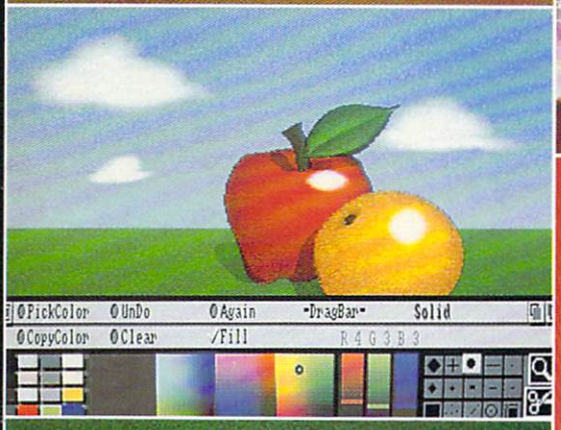
Answer: Anything he wants.

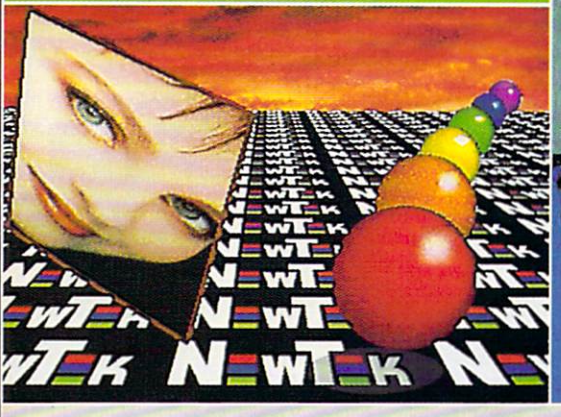
Since FutureSound is IFF compatible (actually three separate formats are supported) your sounds can be used by most Amiga sound applications. With FutureSound and Deluxe Video Construction Set from Electronic Arts, your video creations can use the voice of Mr. Spock, your mother-in-law, or a disturbed super computer.

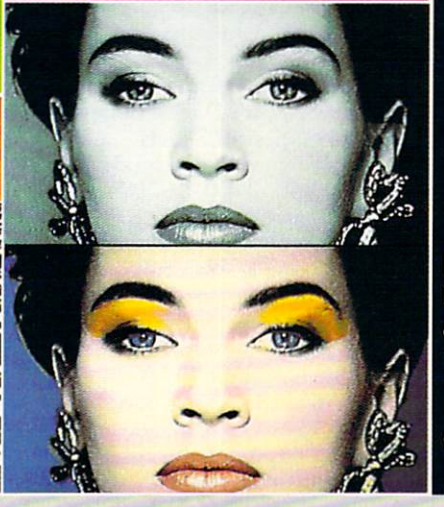
Programming support is also provided. Whether you're a "C" programming wiz or a Sunday afternoon BASIC hacker, all the routines you need are on the non-copy protected diskette.

Your Amiga dealer should have FutureSound in stock. If not, just give us a call and for $175 (VISA, MasterCard or COD) we'll send one right out to you. Ahead warp factor one!

Applied Visions, Inc., Suite 2200, One Kendall Square
Cambridge, MA 02139 (617) 494-5417







ONLY DIGI PAINT CAN DO ALL THIS

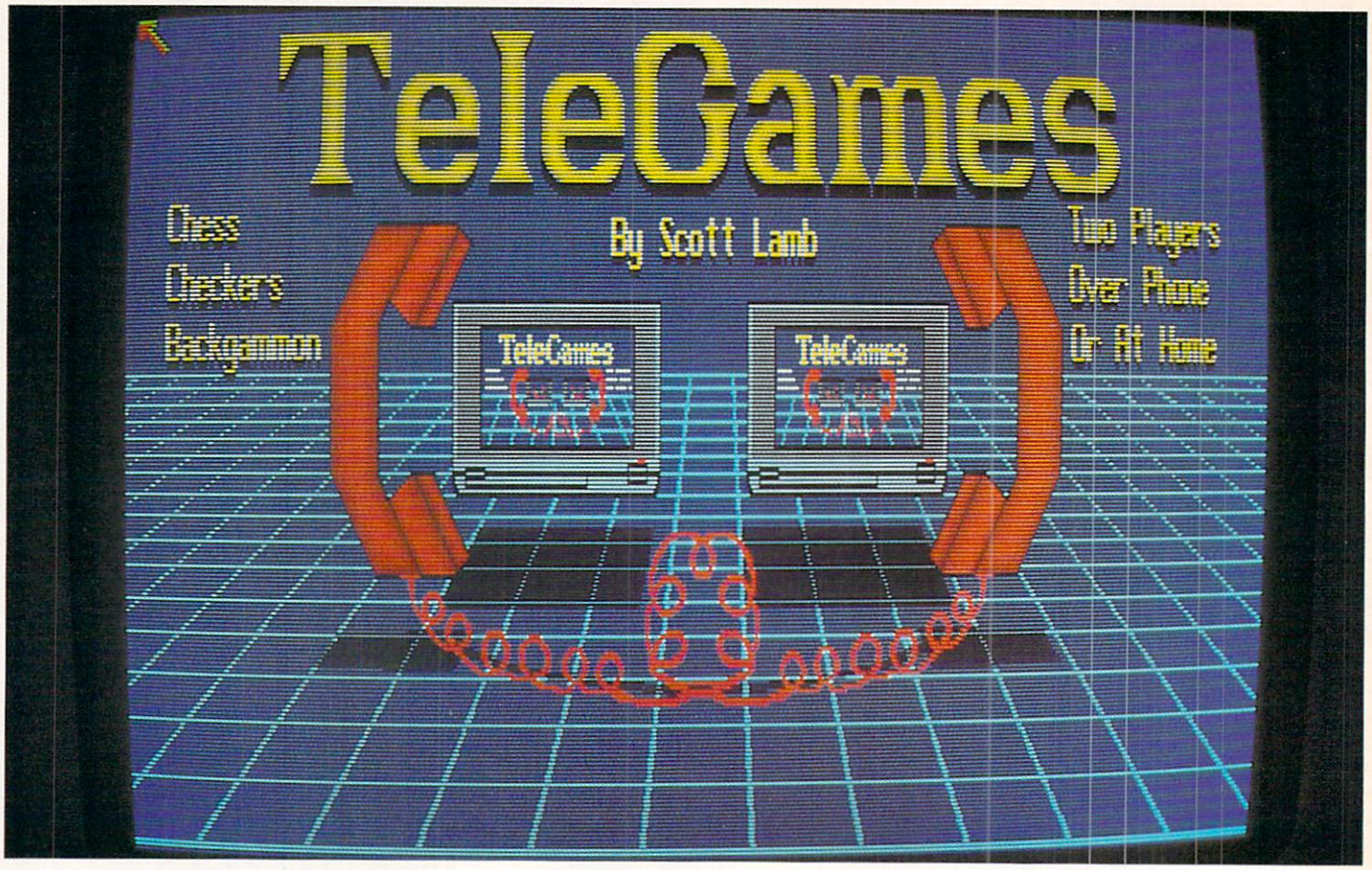
Get the maximum graphics power from your Amiga. Create stunning, lifelike computer artwork with Digi-Paint, the first full-featured 4096 color (Hold and Modify) paint program. Break the "32 color barrier" and finally realize the potential of your Amiga with Digi-Paint's advanced features:

- 4096 colors on screen simultaneously
- NewTek's exclusive enhanced HAM mode
- Dithered HAM gradient fill
- Full screen effects including double, half size, mirror reverse and more
- Full IFF and Digi-View compatibility
- Use 320x200 or HAM hi-res 320x400 resolutions
- Fat bits Magnify mode
- Rectangle, oval, line and other drawing tools
- 12 different paint modes including blending, tinting and smooth shading
- Full lasso cut and paste with automatic edge blending
- Programmed completely in assembly language for fast, smooth response

Find out why Byte Magazine called Digi-Paint "Remarkable". Available now at your local Amiga dealer or call: **1-800-843-8934.**

ONLY $59.95

NEWTEK
INCORPORATED



TeleGames is what you've waited for. The Future is here.

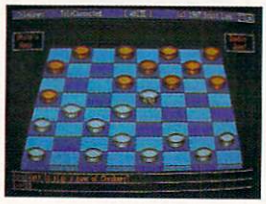
TeleGames allows you to use your computer and modem to play Chess, Checkers and Backgammon with a human opponent over the telephone. Only $34.95!

TeleGames Features

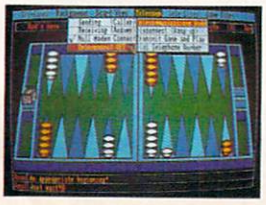
* Chess * Checkers * Backgammon
* Superb Graphic Game Simulations
* Smooth Depth Arranged Movement
* 4 angle 3D & 2D view perspectives
* Digitized Sound Effects
* Compatible with any modem
* 300, 1200, 2400, 9600 Baud
* Call originate or answer
* Null Modem Connect option
* Save Game & Transmit Game options
* Opponent File Directories
* Send and Receive Typed Messages
* Easy to Use Menus & Requesters
* All Official Game Rules Supported
* Play Over the Phone or at Home
* Legal Moves Graphically enacted on the TeleConnected computer
* Fully copyable to hard disks
* Upgrades available on our BBS

3D Chess

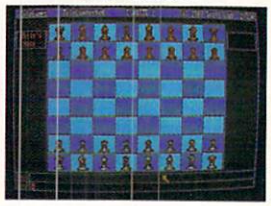
3D Checkers



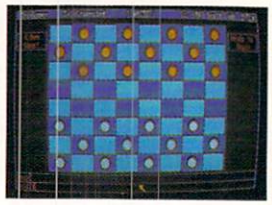
3D Backgammon



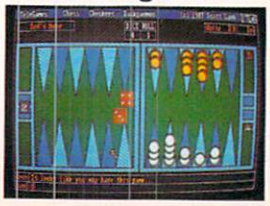
2D Chess



2D Checkers



2D Backgammon



If you Enjoy Telecomputing, You'll Love TeleGames!

Published by Software Terminal
3014 Alta Mere, Fort Worth, TX 76116
817-244-4150 Modem: 817-244-4151
Dealer Inquiries Invited

Amazing Dealers

The following are Amazing Dealers, dedicated to supporting the Commodore-Amiga™. They carry Amazing Computing™, your resource for information on the Amiga™.

If you are not an Amazing Dealer, but would like to become one, contact:

PiM Publications, Inc.
P.O.Box 869
Fall River, MA. 02722
1-617-678-4200

Alabama
Abax Data Systems — Huntsville
Alabama Book Store — Tuscaloosa
Command Computers — Birmingham
Electronics Boutique — Hoover
Madison Books & Computers — Madison
Mail Tel Communications — Mobile
Mel's Photo & Computer Shop — Montgomery
Universal Computer System — Mobile

Arkansas
SIS Inc. — Mt. Home
The Micro Shop — Little Rock

Arizona
Computer West — Phoenix
Computer Works — Glendale
Copperstate Business Systems — Phoenix
Copperstate Computers — Tucson
Hacko's — Tempe
North American Digital — Tucson

California
Aday International — Foster City
Augnet — Freemont
Balboa Office Supply — San Diego
Bits n Pieces Comp Prod. — Anahiem
Brown Knows Computing — Redlands
Byte Factory Computers — Salinas
C.F.T. — Clairemont
Century Computer Systems — La Habra
Circus of Books II — LosAngeles
Commotech — National City
Compare Inc. — Ridgecrest
Computer Attic — Palo Alto
Computer Nook, Inc. — San Bernadino
Computer Outlet — San Diego
Computer Outlet — San Marcus
Computer Time — Citrus Hts
ComputerLand Freemont — Freemont
Datapnile — Watsonville
Delta Spectrum — Laguna Beach
De Lauer's News Agency — Oakland
Electronics Boutique — West Los Angeles
Empire Computers — Santa Rosa
Harding Way News — Stockton
Home Computing Center — Concord
Home Computing Center — Mill Valley
Home Computing Center — San Leandro
Home Computing Center — San Bruno
HT Electronics — Sunnyvale
K J Computers — Granada Hills
Levity — North Hollywood
LTC Computers — Santa Ana
Micro Flight — San Jacinto
Nonagon Software — El Cerrito
Op Amp Tech Books — Los Angeles
World Book & News — Hoolywood
R&R Software — Chula Vista
R&R Software — San Diego
Ridgecrest Computer Ctr. Inc. — Ridgecrest
S.O.S. Computers — Los Angeles
Software First — Santa Rosa
Software Cellar — Anahiem
Software Plus — Citrus Hieghts
Tower Books — Chico
Tower Books — Citrus hights
Tower Books — Concord
Tower Books — Mountain View
Tower Books — Sacramento
Tower Books — San Mateo
The Computer Room — Scotts Valley
Winnar's Circle — Berkley

Colorado
Book Tech Distributing — Englewood
Citadel Computers — Colorado Springs
Computer Discount — Denver
Computer Room — Aurora
Ebert Personal Computers — Aurora
Micro Computer World — Fort Collins
Micro World — Lakewood
Sun Country Computers — Colorado Springs
Aurora Newsstand — Aurora

Connecticut
Amigo Business Computer — Norwalk
Computer Outlet — PlaVernon
Electronics Boutique — Danbury
Friendly Computers — Plantsville
Mnemonics Inc. — Norwalk
Personal Computer center — Norwich
Software City — Orange
Software Kingdom — East Windsor
Spectrum Computers — New Haven

Delaware
Castle Video — Newark

Florida
A A Computers — Jacksonville
Commodore Country Inc. — Pinellas Park
Computer Bar — Pennsacola
Computer Base — Brandon
Computer Image — Miami
Computer Specialty — Melbourne
Computer Terminal Inc. — Sebring
Computers 4 Rent — W. Palm Beach
Cotronics Inc. — Stewart
Education Computers Etc. — Talahasse
Electronics Boutique — Boynton Beach
Electronics Boutique — Tallahassee
Electronics Boutique — Miami
Electronics Boutique — Orlando
Family Computers — Sarasota
Florida Book Store — Gainesville
Games 'N' Gadgets — Orange Park
Games 'N' Gadgets — Jacksonville
Games 'N' Gadgets — Tallahassee
Games 'N' Gadgets — Clearwater
Gulf Coast Computer Exchange — Panama City
Heartland Computers — Mulberry
K&L Computers — E. Palatka
Megaport Computer Center — Venice
Micro Computer Center — Largo
New Age Electronics — St. Petersburg
Random Access Computers Inc. — Ft. Walton Beach
The Open Door — Cocoa
The PC Collection — Coral Gables

Georgia
Electronics Boutique — Atlanta
Electronics Boutique — Atlanta
Electronics Boutique — Kennesaw
Future System — Augusta
Games 'N' Gadgets — Atlanta
Games 'N' Gadgets — Atlanta
Software South Inc. — Savannah
The 64 Store — Atlanta

Hawaii
Astro Computers & Electronics — Aiea
Computer House Inc. — Honolulu
J & C Repair — Aiea

Idaho
A.B.J. Computer & Video — Boise
A.B.J. Computer & Video — Nampa

Illinois
A Brokester — Rockford
ComputerLand O'Hare — Park Ridge
ComputerLand of Niles — Niles
DigitalWorld — Addison
I.C.V. Data Center Inc. — Mundelein
Illini Microcomputers Inc. — Naperville
Interstate Computer Service — LeMont
Kroch's & Brentano's — Chicago
Micro Age — Oaklawn
Software Or Systems — Fairview Hts.
Software Plus — Wheeling
Software Plus West — Hanover Park
The Memory Expension — Normal
Unique Computer — Sterling

Indiana
Burkat Computer Ctr. — South Bend
Bytrex Computer — Ft. Wayne
Bytrex of Angola — Angola
Computer Corner — Fort Wayne
Computer People Inc. — Michigan City
Greg Chaney — Greenfield
Micro Computer Centers Inc. — Indianapolis
Spectra Computer Sys. Inc. — Evansville
Vons Computers — West Lafayette

Iowa
Gromet Computers — Des Moine
Micro Computer Applications LTD — Marshalltown

Kansas
J & L Electronics — Liberal
MidKansas Computers — Newton
Midwest Computers — Manhattan
Throbbred Computers — Topeka

Kentucky
Computer Dimensions Inc. — Lexington
Reality World Computers — Lexington
The Software Source — Louisville

Louisiana
Computer Time — Shreveport
Electronics Boutique — Kenner
Electronics Boutique — Monroe
Electronics Boutique — Laurenceville
Modern Business Machines — Metairie
Software Center International — Metairie
Software Solutions — Baton Rouge
Software Source — Lafayette
The Computer Clinic — Lafayette

Maryland
Buried Treasure — Rockville
Computer Crafters — Wheaton
Computerware TODAY — Gaithersburg
Compuvision Vision Center — Columbia
Electronics Boutique — Baltimore
Electronics Boutique — Baltimore
Electronics Boutique — Anne Arundel County
Electronics Boutique — Owings Mills
Games 'N' Gadgets — Baltimore
Games 'N' Gadgets — Baltimore
Games 'N' Gadgets — Columbia
Layhill Newsstand — Kensington
Micro Computer Center — Baltimore
Software Advantage — Rockville
Software N Things — College Park
Waldorf Computer — Waldorf

Massachusetts
Club Computer — Cambridge
D-Five — Bedford
Electronics Boutique — Holyoke
General Computer Store — Framingham
HCS Computer Center — Marshfield
LCA Video & Computer Ctr. — Norwood
Memory Location — Wellsley
OmniTek Computers — Tewksbury
Pioneer Valley Data Equipment — Amherst
The Bit Bucket — West Newton
The Software Haus — Quincy
Tycom Inc. — Pittsfield

Maine
Computer Barn — Damariscotta

Michigan
Basic Computer Center — Grand Rapids
Bits-Bytes-Nibbles — Petowskey
Canton Computer — Canton
CompuSoft — Kalamazoo
Direct Access — Lansing
Edmunton Computer Center — Pontiac
Galaxy Computers — Allen Park
Global Computer Center — Dearborn
I O Software — Flint
Michigan Software — Novi
Midland Computer Shop — Midland
Central Park Computers — Okemos
Pro-Video — Okemos
Programs Unlimited — Grand Rapids
Roseville Computer Store — Warren
Slipped Disk — Madison Heights
Slipped Disk — Southfield
Software Plus — Grand Rapids
Specialists Computer — Livonia
State Street Computers — Ann Arbor
T.J. Enterprises — Clinton
The Soft Warehouse — Grand Rapids
The Soft Warehouse — Kalamazoo
The Soft Warehouse — Portage
Ye Olde Computer Shoppe — Ypsilanti

Minnesota
Computers, etc. — Eagan
Computer Outfitters — Roseville
Computer Place — Minneapolis
JMH Software — Maple Grove
MCD of Hibbing Inc. — Hibbing
Midwest Computer Services — Clearwater
OnLine Computers — Minnetonka
Specialists IN — Hopkins
Valiant Inc. — Stillwater

Mississippi
Electronics Boutique — Ridgeland
Enterprises Unlimited — Jackson
National Computer Center — Ocean Springs

Missouri
Associated Computer Services — Springfield
Brands Mart Computers — Kansas City
Data Byte Computer Co. — St. Louis
Instant Replay Ltd. — Chesterfield
Systems Plus — Independence
Systems Plus of St. Louis — St. Louis

Montana
Appelgren Computer Systems — Great Falls

Nebraska
BULLS-EYE — Lincoln
My Com Inc. — Plattsmouth
Double E Electronics — Omaha

Nevada
Century 23 Inc. — Las Vegas
Computer World — Las Vegas
ComputerLand of Carson — Carson City
Tower Records — Las Vegas

New Hampshire
Computer Mart of New Hampshire — Nashua
Diversified Computers — Keene
Electronics Boutique — Nashua

New Jersey
Electronics Boutique — Voohnees
Electronics Boutique — Wayne
Electronics Boutique — Rockaway
Electronics Boutique — Cherry Hill
Electronics Boutique — Woodbridge
Family Computer Centres — South Orange
Games 'N' Gadgets — Burlington
Games 'N' Gadgets — Livingston
Morris County Stationers — Chester

New Mexico
New Horizon Computer System — Alamogordo
Page One Newsstand — Albuquerque
Technological Concepts — Las Cruces

New York
Amuse — New York
Amicus- Hudson Valley — Wappingers Falls
Amiga Business Computer — North Port
Byte Shop — Merrick
Computrite Center — White Plains
Computer Outlet — Jamestown
Computer Palace — Felden
Cromland Inc. — Brooklyn
Electronics Boutique — Lake Grove
Electronics Boutique — Huntington Station
Electronics Boutique — Albany
Electronics Boutique — Valley Stream
Games 'N' Gadgets — Nanuet
Games 'N' Gadgets — Garden City
Leigh's Computers — New York
Mnemonics — North Port
Software Centre International — Forest Hills
Software & Such — Scotia
Software City — Forest Hills
Software Supermarket — Kenmore
Star Tech Systems — Massena
Synfaire Trading — New York
Tadiran — New York
Tedrow Business Products — Rochester
The Computer Cellar — Albany
The Working Computer — Stoney Brook
Video Computer Bin — Brooklyn
Video Computer Center — Rome
Video Connection — Schenectady
Village Computer Inc. — New York
World Computers Inc. — Hicksville

North Carolina
Digitz — Raleigh
Games 'N' Gadgets — Raleigh
Software City — Raleigh
Software City — Charlotte
TDS Computers — Durham
Triad Computers — Greensboro

North Dakota
Computer Associates — Fargo
G. F. Computers — Grand Forks
The Computer Store — Minot

Ohio
Computer Network — Newark
Computer Site — Cleveland
Earthlise Micro Systems — Columbus
Electronics Boutique — Columbus
Fairborn Home Computer — Fairborn
Games 'N' Gadgets — North Randall
Lakes Consumer Electronics — Akron
Main Office Supply — Coshocton
Micro Center — Columbus
Microwave Magic — Fairfield
North Coast Programming — Willoughby
Personal Computer Co. — Youngstown
Program Store — Columbus
Quality Computer Appliances — Toledo
Saxtan Computer Centar — Sandusky
Software Centre International — North Olmsted
Software and More — Cincinnati
Software City — Centerville
Software Connection — Mayfield Hts.

Oklahoma
Colonial Video & Computer — Bartlesville
Micro Link Computer Store — Tulsa
P C Tech — Tulsa
Second Hand Software — Oklahoma City
Video-Comp Inc. — Lawton

Oregon
Andromeda — LaGrande
Clackamas Computers — Beaverton
Clackamas Computers — Clackamas
Computron Business Systems — Portland
Software Express — Eugene

Pennsylvania
Alpha Omega EBE — Warren
Basic Computer Systems — Hermitage
Circle Computer — Ephrata
Computers and Games — Reading
Data Softque Computerware — Pittsburg
Downington Computer Center — Downington
East Coast Software — Hershey
Electronics Boutique — North Wales
Electronics Boutique — Langehorne
Electronics Boutique — Exton
Electronics Boutique — Norristown
Electronics Boutique — Pittsburgh
Electronics Boutique — King of Prussia
Electronics Boutique — Whitehall
Games 'N' Gadgets — Lancaster
Games 'N' Gadgets — King of Prussia
Games 'N' Gadgets — Media
Games 'N' Gadgets — West Mifflin
Genel's Books — King of Prussia
North Central Digital Systems — Danville
Pittsburgh Computer Store — Pittsburgh
The Electronic Boutique — Norristown
Triangle Computers — Indiana

Rhode Island
International Computer Services — Johnston
Micro Limits — Smithfield
Micro Limits — Warwick

South Carolina
Avcom International Inc. — West Columbia
Electronics Boutique — Greenville
Horizon — Greenville
Micro Computer Depot — Sumter
Watson's Jewlery — Chesterfield

South Dakota
Computer Business Center — Rapid City

Tennessee
ComputerLab of Memphis — Memphis
Games 'N' Gadgets — Memphis
Software First — Nashville

Texas
Byte Shop — Austin
C.E.M. Corporation — Brazoria
Colonial Video — Houston
Computer Age — Houston
Computer Age — Dallas
Computer Magic — Austin
Computer Revelations — Houston
Computer Time Inc. — Marshall
FNS Computer Services — Texarkana
Metropolitan Computer — Richardson
Micro Search — Houston
Softex — Watauga
Software Library — Wichita Falls
Software Plus — El Paso
Software Terminal — Fort Worth
The Computer Experience — San Antonio
The Computer Stop — Abilene

Utah
Computers Plus — Murray

Vermont
Ray Supply Inc. — South Burlington

Virginia
Colony Ltd. — Hampton
Diskovery — Falls Church
Family Computer Center — Fairfax
Games 'N' Gadgets — Hampton
Games 'N' Gadgets — Virginia Beach
Games 'N' Gadgets — Virginia Beach
Software Central — Norfolk
Software City — Richmond
Virginia Micro Systems — Woodbridge

Washington
A.P.P.L.E. CO-OP — Renton
Apple Valley Computers — E. Wenatchee
Bull Dog News — Seattle
Byte World — Kent
Com-Soft Computers — Everett
Computers + Inc. — Tacoma
Computer Outlet — Edmonds
Family Computers — Lynwood
Kitsap Amiga Operator System — Bellview
Micro Age Computer Store — Bellview
Micro Specs Inc. — Spokane
Nibble Shs — Tocoma
Omni International — Seattle
Programs Plus — Seattle
Take-A-Byte — Kennewick

West Virginia
Layhill Newsstand — Charleston

Wisconsin
Colortron Computers — Racine
Computer Software Center — Milwaukee
Fox Valley Personal Computer — Oshkosh
Mom's Computers — Stevens Point
Software Shoppe — Fond Du Lac
TMW Software — Greenbay
TMW Software Inc. — Wausau

Wyoming
Byte Emporium — Laramie

Canada
Compute or Play — Alberta
Computerworks — Alberta
Computer Shop — Alberta
Data Pro — Alberta
Computer Shop of Calgary — Alberta
The Software House — Alberta
Hindsin Computing — Alberta
Red Deer Micro Systems — Alberta
Memory Land — Alberta
Computer King — British Columbia
Genl — British Columbia
Nutec — British Columbia
Sprita — British Columbia
Strider — British Columbia
Super Software — British Columbia
Canadian Computer Era — Manitoba
Micromart — Manitoba
ComputerWorld — Manitoba
J.W. Allen — Manitoba
Kobetsk — Manitoba
Minicomp — Manitoba
Arkon — Ontario
Chatam Computer — Ontario
Computerland — Ontario
Comspec — Ontario
Electronics 2001 — Ontario
Hastings Data — Ontario
The Computer & You — Ontario
Thornhill — Ontario
Software Co. — Quebec
Deltronic Solare — Quebec
Mierodata — Quebec
Info Mereler — Quebec
Laval Micro — Quebec
Vision Tronique — Quebec
Distribution Visipro — Quebec

Distributed in Canada by
Avita Software Distributors — Concord, Ontario
Compulit Distributors — British Columbia
Phase 4 Dist. Inc. — Calgary, Alberta

Argentina
Logic Computer — Buenos Aires

Australia
High Technology Computer Supplies — Brighton, Melborn
Maxwell Office Equipment — Abbotsford, Melburn

Japan
Pineapple 6502 — Tokyo, 111

Saudia Arabia
Abdulla Fouad & Sons — Daman

Switzerland
Software Land AG — Zuerich

Amazing Computing™ is also available in most B. Dalton Booksellers stores, B. Dalton Software stores, and Software Etc. locations.

Amazing COMPUTING™

Publisher:	*Joyce Hicks*
Circulation Manager:	*Doris Gamble*
Asst. to the Publisher:	*Robert James Hicks*
Corporate Trainer:	*Virginia Terry Hicks*
Traffic Manager:	*Robert Gamble*
Managing Editor:	*Don Hicks*
Submissions Editor:	*Ernest P. Viveiros Jr.*
Hardware Editor:	*Ernest P. Viveiros Sr.*
Amicus & Tech. Editor:	*John Foust*
Music & Sound Editor:	*Richard Rae*
Art Director:	*Keith M. Conforti*
Advertising Manager:	*John D. Fastino*
Copy Editor:	*Michael T. Cabral*
Production Manager:	*Mark Thibault*
Assistant PM	*Keven P. Desmarais*

Advertising Sales & Editorial
1-617-678-4200

Special thanks to:
Lynn Hathaway
Donna Peladeau
Traci Desmarais
Pilar Medeiros
Donna Thibault
Betsy Piper at Tech Plus.
& Paul Boden at
Software Supermarket

Amazing Computing™ (ISSN 0886-9480) is published by PiM Publications, Inc., P.O. Box 869, Fall River, MA 02722. Subscriptions: in the U.S.12 issues for $24.00; in Canada & Mexico, $30.00; Overseas; $35.00. Printed in the U.S.A.

First Class or Air Mail rates available upon request.

PiM Publications, Inc. maintains the right to refuse any advertising.

PiM Publications, Inc. is not obligated to return unsolicited materials. All materials requesting return must be received with a Self Addressed Stamped Mailer.

Amazing COMPUTING™

Volume 2, Number 10

CONTENTS

Amazing Features

Amazing Columns

Amazing Reviews

Amazing Programming

Amazing Departments

From The Editor:

Amazing Computing™ in Color?
Color? In Amazing Computing™? Wait, what goes here?

It may come as a surprise to our readers that this issue has color on the inside of the magazine. *Amazing Computing™* has produced eighteen issues of continually growing Amiga information without the use of color. Some readers may wonder why we think we need color now.

To be honest, it would be less expensive to continue *Amazing Computing™* with an all black and white interior. There are two important reasons for using color:

- The Amiga is a great graphics machine and we need to demonstrate its color potential. Our readers will never know how many *Amazing Computing™* issues were created with great color pictures. The photos were pasted to copy sheets and looked terrific, however, we could only settle for a black and white halftone in the completed magazine.

- We must also fill the need of our advertisers, who require color advertising to best showcase their products.

OH NO, NOT ADVERTISING!
We constantly receive letters from well-meaning readers who beg us not to change a single thing about the magazine. They are apparently pleased with programming information and reports . . . they are even tolerant of our coverage of "softer" subjects such as trade shows!

I understand our readers' concern. They are happy with the way *Amazing Computing™* has been covering the Amiga™ and they do not want change. Readers are very wary of change. An inability to seperate advertising content from editorial content sticks out in the reader's mind. Perhaps readers feel this way because some publications have a lot more advertising and a lot less substance (our reader's choice of phrase, not mine) than *Amazing Computing™*.

However, the key to *Amazing Computing™'s* substance is our dedicated readers.

Our Past
In the past nineteen issues of *Amazing Computing™*, AC has delivered almost 1800 pages. Of these pages, more than 72% were text (or, if you prefer, less than 28% were advertising). Each issue has been an improvement over the previous issue. We are proud of this heritage.

We are as proud of you, our readers, as we are of our heritage. To be completely honest, this solid, growing body of Amiga information is the result of Amiga enthusiasts all over the world. Our readers supply most of the written copy for *Amazing Computing™*. We rely completely on the creativity of our readership. We are constantly searching for the new, different approaches our readers use in working with their Amigas.

Our Present
We will continue to rely on you as our main resource. It is through your submissions that *Amazing Computing™* has been able to maintain this level of highly interesting material. It is through your eyes that we see the Amiga market. *Amazing Computing™* has always been a forum for reader comments and suggestions: maintain this enthusiasm and we will continue to produce the magazine you prefer.

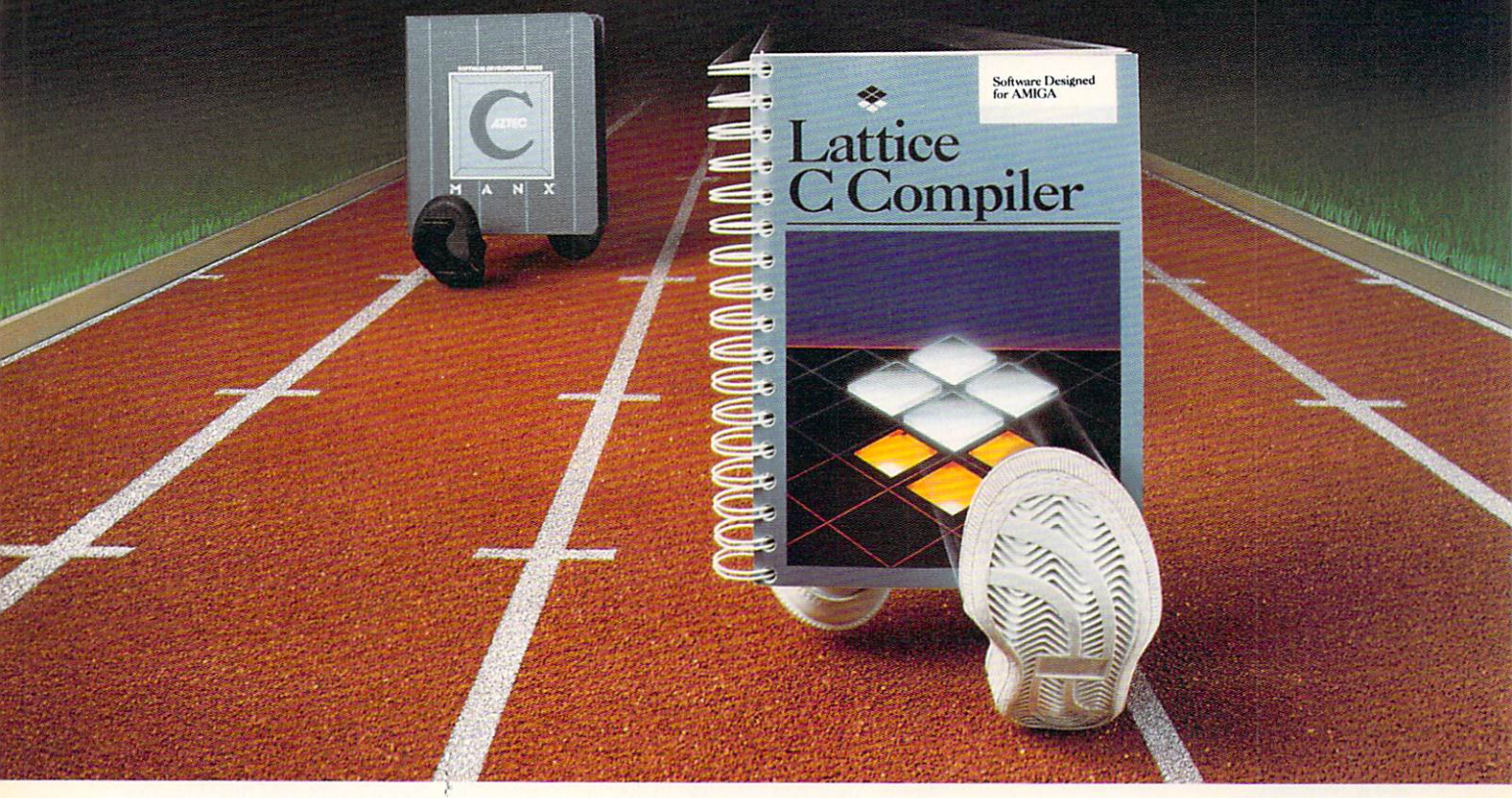
Also, advertising can play a large part in the developement of the Amiga. Many of us see developer's new products for the first time through advertising. If the individual companies require COLOR in their advertising, AC should grow to satisfy this need. It is important for the growth of the entire Amiga community to allow all advertisers an access to the market.

Developers must also feel that the Amiga market is secure enough to maintain several full color magazines. This is political, since in most cases, a black and white story can read the same in any magazine. COLOR, however, allows the Amiga to shine where it shines the best, through its powerful graphics ability.

In short, all products must continue to improve themselves in order to remain marketable — including *Amazing Computing™*. We promise to do all we can with our newest tool, COLOR. Bear with us, as we spread our wings.

Don Hicks
Managing Editor

C who's winning the race. Lattice C for Amiga.



Lattice C has long been recognized as the best C compiler. And now our new version 4.0 for Amiga™ increases our lead past the competition even further.

Ready, set, go. The new Lattice AmigaDOS C Compiler gives you faster, more efficient code generation and support for 16 or 32-bit integers. There's direct, in-line interface to all Amiga ROM functions with parameters passed in registers. What's more, the assembler is fully compatible with Amiga assembler syntax.

More great strides. The linker, Blink, has been significantly enhanced and provides true overlay support and interactive recovery from undefined symbols. And you'll have a faster compile and link cycle with support for pre-linking.

	Lattice® Version 4.0	Manx® Version 3.40
Dhrystone	1294 Dhrystones/second	1010 Dhrystones/second
Float	22.20 Secs. (IEEE Format) 10.16 Secs. (FFP Format)	98.85 Secs. (IEEE Format) 17.60 Secs. (FFP Format)
Savage (IEEE)	47.67 Secs./.000000318 Accuracy	119.6 Secs./.000109 Accuracy

There's no contest. Standard benchmark studies show Lattice to be the superior C language development environment. With stats like these, it's no wonder that Commodore-Amiga has selected Lattice C as the official Amiga development language.

Going the distance. You'll experience unsurpassed power and flexibility when you choose from several cost-effective development packages. There is even a full range of supporting products, including a symbolic debugger, resource editor, utilities and specialized libraries.

You'll discover that your software purchase is backed by an excellent warranty and skilled technical support staff. You'll appreciate having access to LBBS—one of the world's first 9600 baud, 24-hour bulletin board services. And you'll be able to conference with other Lattice users through the Byte Information Exchange (BIX) network.

Cross the finish line. Order your copy of the Lattice AmigaDOS C Compiler today. We'll supply the speed. You bring the running shoes.

Lattice
Subsidiary of SAS Institute Inc.

Lattice, Incorporated
2500 S. Highland Avenue
Lombard, IL 60148
Phone: 800/533-3577
In Illinois: 312/916-1600

Amazing Mail:

Dear AC:
I will appreciate it if you would print the following notice in the Amicus Network section of *Amazing Computing*™. If you feel that this notice belongs on another section of the magazine, please forward it to the proper editor.

Your help is appreciated.

Mexico City Amiga Users Group

I am interested in forming an Amiga Users Group in Mexico City. If you are interested in participating, please call me at: 551-18-04 or 760-92-41 Monday through Saturday before 6 PM or write to:

Miguel A. Romero
Norte 170 No. 529
Col. Pensador Mexicano
15510 Mexico, DF

The purpose of the group is to facilitate the exchange of information, software and expertise among owners and users of the Amiga computer. Your suggestions for these goals and any others are most welcome.

Thanks
Miguel A. Romero

Good luck and keep us informed of your progress.

Any inroad the Amiga can make means a larger machine base, and will only yield more and better third party support.

Dear AC:
I am amazed.

If I had not seen the article entitled "Skinny C Programs" in your *Amazing Computing*™ Volume 2.8, I would not have believed that you would have published such an outdated and biased article. The lead in sentence was so biased that I thought I was reading an article that was submitted by a competitor of Lattice C. Two of my friends are Amiga and Lattice C owners, and both are satisfied with the product. Thus the lead in sentence from the article:

"Like many other Amiga users, I have been less than satisfied with the Amiga Lattice C compiler"

negates any redeeming value that article may otherwise contain.

Further, any article that talks about the deficiencies of an old product does not deserve the space in a magazine as fine (I thought) as yours. As you know, the product discussed in the article has been replaced by an enhanced version. I was a user of Lattice C 3.03 for about a year, but have had the improved 3.10 version for several months now. I was pleased with the product since the earlier version. The new version has only increased my product satisfaction.

In addition, the technical details explained in this article are already available in the Lattice C documentation, and in the Amiga Technical Reference Manuals, so I gained no information from reading the article.

Please get back on the track, and publish meaningful, factual, unbiased articles that will be informative to the Amiga user community, including both programmer and non-programmer users.

Sincerely,
Larry A. Black

Pleasing every Amiga user with every AC article is quite a tall order. Not everyone has the same viewpoints and not everyone has the same difficulties.

The article referred to above ("Skinny C Programs," V 2.8) was not intended to denigrate the Lattice C Compiler or its dedicated following of successful users. Rather, the tips offered were targeted at those users who were experiencing some problems. Sure, to veteran Lattice users, the article may have lacked useful information . . . but, for the struggling user, the article may have supplied the missing pieces to a frustrating puzzle.

We are aware that the new Lattice version 3.10 has been on the shelf for some time now — in fact, an in-depth review of the new version was printed in the very next issue (September, V 2.9, p. 51). We had been developing that review for some time, but still felt it necessary to print the "Skinny C" article for those users who were still working with 3.03. A release of an updated version of a product does not prompt us to simply drop our coverage of the previous version. Many users in the wide-ranging Amiga pool do not have the latest release. We realize this fact and try to give equal space and time to ALL AC readers.

We do, however, welcome the negative feedback. The many, many positive and complementary letters we receive are warmly appreciated, but the few letters of dissatisfaction are also a great help to us. In order to improve AC from a reader's point of view, we need you to point out our weak spots. If you run across a deficiency, please don't hesitate to write.

Hi Guys & Gals:
Congratulations on an outstanding new logo! A masthead you can truly be proud of! Kudos to Art Director, Keith Conforti! A first class logo for a first class magazine!

This signifies, in my opinion, the coming of age of *Amazing Computing*. The maturing of a brash new member of the publishing community into a mature respected community leader!

Keep the good stuff coming! I enjoy every page - the smell of the ink - the quality of the paper - but most of all, the comments, the rumors and the reportage of trade shows.

You do it like no one else!

Richard Crommett
Keller, TX

Thank you for the encouragement. I hope you like the addition of color this month. If it seems we are becoming too "polished," please let us know.

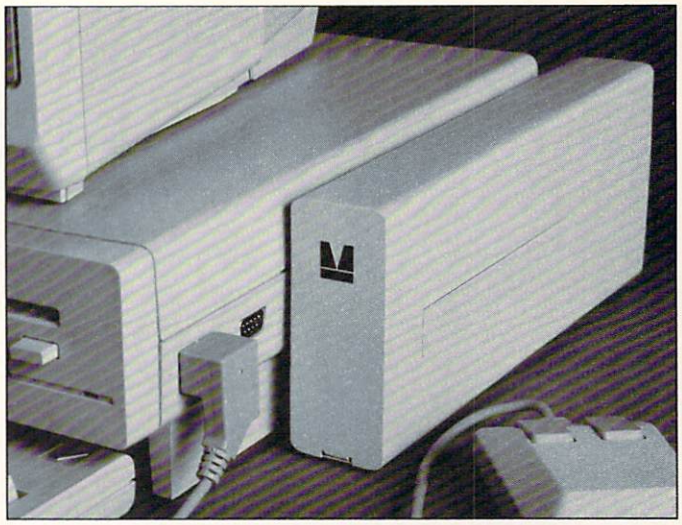
•AC•

AC encourages your comments, good or bad. Please send us your concerns. Through your feedback, AC remains your magazine. •AC•

AVAILABLE NOW!

StarBoard2

Auto-Configuring Fast RAM
Zero Wait States
User Expandable from 512k to 2 Megabytes
Bus Pass-Through
MultiFunction Option: battery/ clock, FPU, parity, Sticky-Disk



If you've owned your Amiga® for a while now, you *know* you definitely need more than 512k of memory. You probably need *at least* double that amount...but you might need as much as an additional two megabytes. We want to urge you to use **StarBoard2** as the solution to your memory expansion problem –and to some of your other Amiga-expansion needs as well!

It's small, but it's BIG–

Since most of you want to expand your Amiga's memory without having to also expand your computer table, we designed **StarBoard2** and its two optional "daughterboards" to fit into a sleek, unobtrusive Amiga-styled case that snugly fastens to your computer with two precision-machined jackscrews.

The sculpted steel case of **StarBoard2** measures only 1.6" wide by 4.3" high by 10.2" long. You can access the inside of the case by removing just two small screws on the bottom and pulling it apart. We make **StarBoard2** easy to get into so that you or your dealer can expand it by installing up to one megabyte of RAM on the standard **StarBoard2** or up to two megabytes by adding in an Upper Deck.

This card has decks!

The basic **StarBoard2** starts out as a one megabyte memory space with 0k, 512k, or one megabyte installed. If you add in an optional **Upper Deck** (which plugs onto the Main Board inside the case) you bring **StarBoard2** up to its full two megabyte potential. You can buy your **StarBoard2** with the Upper Deck (populated or unpopulated) or buy the Upper Deck later as your need for memory grows.

And you can add other functions to **StarBoard2** by plugging in its second optional deck –the Multifunction Module!

StarBoard2: functions five!

If we count Fast Memory as one function, the addition of the **MultiFunction Module** brings the total up to five!

THE CLOCK FUNCTION:

Whenever you boot your Amiga you have to tell it what time it is! Add a MultiFunction Module to your **StarBoard2** and you can hand that tedious task to the battery-backed, real-time clock/calendar. A small piece of MicroBotics software in your WorkBench Startup-Sequence reads the clock and automatically sets the time and date in your Amiga. And the battery *is* included (we designed it to use an inexpensive, standard AAA battery which will last at least two years before needing replacement).

THE FLOATING POINT FUNCTION:

If any one aspect most characterizes the Amiga it's *fast* graphics! Most graphic routines make heavy use of the Amiga Floating Point Library. Replacing this library with the one we give you with your MultiFunction Module and installing a separately purchased Motorola 68881 FPU chip in the socket provided by the Module will speed up these math operations from 5 to 40 times! And if you write your own software, you can directly address this chip for increased speed in integer arithmetic operations in addition to floating point math.

THE PARITY CHECKING FUNCTION:

If you install an additional ninth RAM chip for every eight in your **StarBoard2**, then you can enable *parity checking*. Parity checking will alert you (with a bus-error message) in the event of any data corruption in **StarBoard2**'s memory space. So what good is it to know that your data's messed up if the hardware can't fix it for you? It will warn you against saving that data to disk and possibly destroying your database or your massive spreadsheet. The more memory you have in your system the more likely it is, statistically, that random errors will occur. Parity checking gives you some protection from this threat to your data residing in Fast RAM. Note that the Amiga's "chip" RAM cannot be parity checked.

THE IMMORTAL MEMORY DISK FUNCTION (STICKY-DISK):

When you've got a lot of RAM, you can make nice big RAM-Disks and speed up your Amiga's operations a lot! But there's one bad thing about RAM-Disks: they go away when you re-boot your machine. Sticky-Disk solves that problem for you. It turns all of the memory space inside a single **StarBoard2** into a Memory Disk that will survive a warm-reboot! When your Amiga attempts to grab a **StarBoard2** in Sticky-Disk mode, a hardware signal prevents the system from acquiring the **StarBoard2** as FastRAM (and thereby erasing your files) –instead it is re-recognized as a Memory Disk and its contents are preserved intact. If you want to work rapidly with large files of data that are being constantly updated (such as when developing software) you can appreciate the Sticky-Disk!

Fast RAM –no waiting!

StarBoard2 is a *totally* engineered product. It is a ZERO WAIT-STATE design, auto-configuring under AmigaDOS 1.2 as Fast RAM. Since AmigaDOS 1.1 doesn't support autoconfiguration, we also give you the software to configure memory in 1.1.

Any applications software which "looks" for Fast RAM will "find" **StarBoard2.** And you'll find that your applications run more efficiently due to **StarBoard2** on the bus.

A passing bus? Indeed!

What good is an Expansion Bus if it hits a dead end, as with some memory cards? Not much, we think –that's why we carefully and compatibly passed through the bus so you could attach other devices onto your Amiga (including another **StarBoard2**, of course!).

The sum of the parts...

A really nice feature of the **StarBoard2** system is that you can buy exactly what you need now without closing off your options for future exapansion. You can even buy a 0k **StarBoard2** (with a one megabyte capacity) and populate it with your own RAM (commonly available 256k by 1 by 150ns memory chips). When you add **StarBoard2** to your Amiga you have a powerful hardware combination, superior to any single-user micro on the market. See your Authorized Amiga Dealer today and ask for **StarBoard2**

SUGGESTED RETAIL PRICING:	
StarBoard2, 0k (1 meg space):	$349
StarBoard2, 0k (2 meg space):	$395
StarBoard2, 512k (1 meg space):	$495
StarBoard2, 1 meg (1 meg space)	$595
StarBoard2, 2 megs installed:	$879
StarBoard2, 2 megs & MultiFunction:	$959
Upper Deck, 0k (1 meg space):	$ 99
MultiFunction Module:	$ 99
also available:	
Standard 256k memory card:	$129
MAS-Drive20, 20 meg harddisk:	$1495
MouseTime, mouseport clock:	$ 50

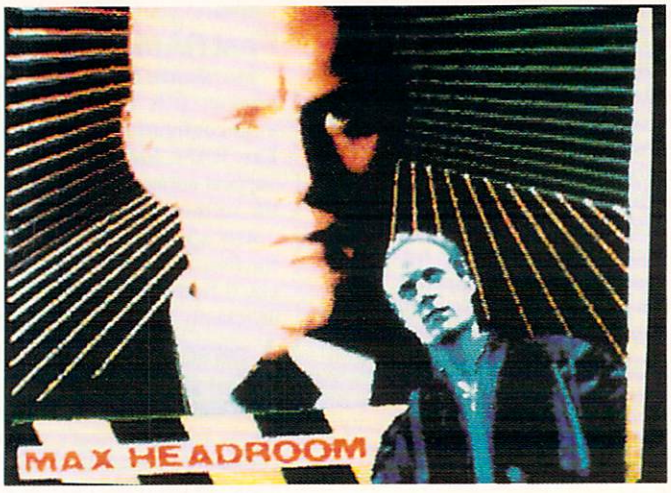
MicroBotics, Inc.
811 Alpha Drive, Suite 335, Richardson, Texas 75081 / (214) 437-5330

Max Headroom
and the Amiga

by John Foust

Inside the East Gate of Lorimar TelePictures in Culver City, California is a walk-up office of the ABC television series "Max Headroom." In one room, the walls are covered with blueprints of the set and detailed drawings of futuristic devices. The floor is littered with real-world devices from the past – an old Heathkit oscilloscope and a "condenser tester."

One desk is piled with papers and videotapes – Fritz Lang's Metropolis, Bladerunner, Brazil, the Max Headroom pilot episodes – and an Amiga computer system. This desk is inhabited by Jeff Bruette, a technical consultant for the Max Headroom series. The Amiga is an integral part of the production of this futuristic series.



"Max Headroom: 20 Minutes Into the Future" piloted six episodes last spring on ABC, and now joins the ABC fall lineup with twenty-two new episodes. The series focuses on the actions of an investigative reporter for a future television network called Network 23. The main character, Edison Carter, was injured in a motorcycle crash while investigating corruption within the Network itself.

Network executives scan Carter's brain to find out what he has learned. A computer expert uses this opportunity to test his latest invention, a method of storing a subject's memory and generating his image. The last thing Carter saw was a low-clearance barrier marked "Max Headroom," thus the name of the computer-generated character who bears a stunning resemblance to Carter.

The stuttering Headroom character stars in a series of commercials for new Coke, as well as hosting a talk show on the Cinemax cable network. A Newsweek cover earlier this year highlighted Max's escalating popularity.

Bruette is also working on an upcoming series called "Secrets and Mysteries." This work is being done in conjunction with Bruette's freelance graphics company, Prism Computer Graphics. He describes the series as "like 'In Search Of', but in a Jules Verne setting."

"Secrets and Mysteries" is a syndicated series produced by ABC. Bruette produces computer graphics for the series with the Amiga. Most of the show's graphics will be done with the Amiga. Computer artist Cris Palamino is working with Bruette to generate images for the show. Bruette describes her work as "meticulous."

"Well, in my own humble, little way I suppose I'm David Letterman, MTV, and Dr. Ruth all rolled into three.

If I do have a fault, it's a very unusual skin condition... it's called perfect"

—Max Headroom

High low tech

Designers on the Max Headroom set study videotapes of futuristic movies, such as Bladerunner, looking for ideas for the Max Headroom set. Bruette describes it as the look of "high low tech, or antique equipment being updated." Some sets mix old electronic equipment with futuristic gear. "This is high tech; the low tech is boring. So, what they [Headroom designers] are trying to do, is take the high tech which everyone enjoys and put a new twist to it."

The characters in the future have a different attitude about technology, too. Bruette explains their perspective.

"Theoretically, all this technology has reached a maximum point. For some reason, the cause for developing all this

almost disappeared, as if one civilization ended and a new civilization moved in. We're into the future, past their discovery stage. They figure out that if you hook up this old Underwood typewriter with a series of sensors on the hammers that send signals to the computer, the computer displays something accordingly. Everybody knows that."

Amiga overlays

For the new series, the Amiga produces the overlay graphics for the ViewPhone, the VidiCam, InterCam and SecuriCam. When two characters are on the phone in the show, a telephone transmits a video image, as well as voice. On both ends of the call, the character sees the name and phone number of the other person. The VidiCam is Edison Carter's portable video camera, used for live updates to the Network. The SecuriCam takes the Big Brother viewpoint, spying on everything that moves.

Bruette uses high resolution Deluxe Paint II to draw the overlay graphics. Large fonts from a custom font disk are used for the lettering. In some cases, letters must appear in succession, or perhaps an indicator on the screen should blink. Bruette uses a preliminary version of a program from Aegis called GrabANIM for this work. GrabANIM is essentially an Amiga stepframe recorder. Invoked with a hot-key sequence, the program looks at the current screen and stores it as a frame of animation, using the new IFF ANIM format. ANIM only stores the differences between successive frames (It only stores the new letter or the absence or presence of the blinking indicator), so file storage is at a minimum.

By recording the Deluxe Paint II screen as the letters emerge, Bruette creates an entirely computer-controlled animation sequence that can be transferred to video tape (Aegis has released a freely distributable program called ShowANIM that plays ANIM animations).

Many companies, including Electronic Arts, Aegis Development and Byte-by-Byte, have supplied early releases of their software. According to Bruette, the companies will not be mentioned in the closing credits, but that it is always advantageous to be part of a popular show.

Amiga vs. IBM

For simple overlays like the SecuriCam, you may wonder why the Amiga is better than the equipment used in the early episodes? For the pilot episodes of Max Headroom, the overlays and other video special effects were accomplished with an IBM-PC-based system equipped with a special genlock board. A keyed switcher and graphics from a Chyron video effects machine were also incorporated. There were only two computer systems in use, one on stage and one in the production area.

Richard Lewis is the production designer for the Max Headroom series and also worked with Bruette on the Amazing Stories episode. Lewis explained why the Amiga is an improvement over the Video Image system. "That system had limitations. That's why I was trying to get us to use this system. The IBM system had a palette with a 256 color range, sixteen colors at a time. We never got shades we were happy with. Using the Chyron, you'd lose quality in the overlay by reducing the resolution on videotape cassette."

With the Amiga, much more control is kept in-house. Many video special effects houses have expensive systems for computer graphics. In addition to Prism, the Max Headroom series is currently working with the Post Group, a local video special effects house. The Post Group works with Amigas, as well as much more expensive video computers.

The older, more expensive video effects machines were always a bottleneck, Lewis said. "They have a lot of big toys to play with. When you've spent a quarter-million dollars on a system, you have only one. No matter how fast or sophisticated it is, it is a bottleneck. With the level of stuff we're doing, it doesn't require a quarter-million-dollar machine to do a simple overlay. You can dedicate those machines to a higher level of graphics, instead of turning out everything through the big machine."

The Amiga is a low-cost solution to the bottleneck. Several members of the crew have their own machines. Down the hall, series art director Frank Pezza uses an Amiga and Deluxe Paint to design logos and test colors for the sets and computer graphics. He uses Scribble! to write budgets and reports. Pezza was also art director for Miami Vice, and worked with Lewis on the series "Whiz Kids." He has also done computer graphics design at the Post Group.

Peter Wagg, executive producer, has an Amiga in his office, so he can approve everything that goes on the screen, including any special overlay graphics. Bruette described what happens in Wagg's office. "We go in there and stick a disk in, and say 'Here's what we're using,' and he usually says 'No'." He has something in his mind of what he wants. When something is presented to him, he thinks of something else. It is inevitable that whatever is done, is changed."

Eight miles from Lorimar, the Post Group is getting backed up on the special effects they are producing. Bruette expects the Amiga to take up the slack and contribute more to the show as each episode is produced. "The Amiga is going to end up doing more and more on the show," going beyond overlays, doing such things as network logos. Designers at the Post Group have Amigas "because of us," according to Bruette. Bruette is constantly asked where and how Amigas can be purchased.

Amiga is "too good"

The quality of Amiga video has been sufficent, so far. In fact, production

continued...

people have had to "dirty up" the video, to make it look worse. According to Lewis, "So far, the biggest complaint about the graphics was that they weren't crude enough. Now, at the Post Group, they are using an ADO to shrink it down, [they] then have a camera aimed at the screen to re-photograph it, to distress the video signal. It is a lot of work to put it [the video] through a blender and chop it up. It is the hardest thing for people in video because they are geared to the glossiest, brightest picture you can give – which isn't always what we are looking for."

In one instance, a fault in a genlock combined with the Amiga composite video signal to make vertical bars in the final image. All along, Lewis warned Bruette that he would need to correct the genlock to remove the bars before filming. When producer Peter Wagg saw the bars, though, he asked Bruette to enhance them, to make them more visible. "When Jeff and I did the Amazing Stories segment, there was a reason I was pushing real hard to use the Amiga and digitizing."

Lewis explained why the Amiga was preferable. "We looked at the stuff the video houses could deliver. We had a lot of resistance from the video post-production people at Universal because they all wanted to do it the old way: You shoot the actor, you do some kind of effects on him and play it back. Every time we looked at their effects, it looked like the things you see on sports on ABC. It is very slick and blenderized. Everybody has seen it. If you are trying to say this dude is in the computer, and you're seeing the same thing you see behind a sports announcer, it doesn't click. The Live! board gave us that edge. Like the time Live! takes to update an image when a head turned – you got real interesting effects that said, this is a computer, this is not a trick."

Script to screen

Commodore has worked closely with Bruette in providing computers for the staff of Max Headroom. There are a total of thirteen Amiga systems in use at present, seven Amiga 2000s and six Amiga 1000s. The writers for the show use Commodore PC-10 MS-DOS computers.

"To some degree, Commodore is involved in Max Headroom from 'script to screen," said Bruette, laughing at the overused expression, but adding, "it's really true." Unlike the obstacles Bruette encountered when he asked for genlocks for Amazing Stories, Commodore was very helpful in getting Amiga systems for Max Headroom.

Bruette wanted Amiga 2000 systems because they are more expandable than the Amiga 1000, but Commodore was unable to deliver them in mid-August. "They don't have them to give. You can't get blood from a turnip. It's not that Commodore didn't come through." To Bruette's surprise, Commodore worked things out and sent the Amiga 1000 systems, as well as several Amiga 2000 systems direct from Germany.

Among the devices expanding the Amiga 1000 systems are Commodore genlocks, Xebec hard disks, Microbotics StarBoard II memory boards and a Calcomp color printer. A CSA Turbo Amiga 68020 system had just arrived, and Bruette planned to use it to make VideoScape and Sculpt 3D much faster.

Bruette hopes to expand each Amiga 2000 system with hard disks, dual floppy drives, a genlock board, five megabytes of RAM and a Bridge card as soon as possible. Commodore sent one-drive systems with one megabyte of memory. Since then, he has been sent several two megabyte memory boards and a hard disk controller card.

Bruette attributes much of Commodore's involvement to Commodore's southern California representative, Hal Lafferty of Jack Carter Associates. "He is a great facilitator. He has been extremely helpful on all things so far. If we needed anything right now, and if he had it, it would be over here before the day's end."

For their contribution, Commodore will get a mention in the closing credits in the form of this message: "Production computers supplied by Commodore Business Machines, Amiga Division."

Amigas on the set

Bruette walked through the back lots of Lorimar to the set of Max Headroom. He passed the set of "Knots Landing" on the way, down Garland Avenue. Lorimar also produces the series "Dallas" and "Our House," and movies such as, "The Boy Who Could Fly" and "Perfect Strangers." Lorimar's studios were formerly part of Metro Goldwin Mayer.

Max Headroom production has a building of its own. Each stage building could enclose several small houses. The ceilings are nearly a hundred feet high. The inside is very dark, and the air contains wisps of artifical fog used to enhance the lighting of the sets. Thick batting covers the walls to reduce sound echoes. Each room in every building in the series is built out in the open in the actual building. The Network boardroom is only a few yards from Theora's apartment, for example.

Bryce Lynch's computer lab has a false hallway that only goes back about ten feet, but with mirrors and tricks of "forced" perspective in the set construction; it appears to go on and on to other offices. Up close, the details of each set are reminders of reality, while on television, the illusion of the future is created.

For example, the pink bus used as Blank Reg's Big Time T.V. network headquarters is only a metal shell. The real bus used for exterior shots is stored on a back lot. The walls of the bus interior carry more reminders of reality. Most sets have piles of anonymous old electronic equipment, but up close, you can see that the equipment is actually only infrared thermometers and television calibration equipment. On screen, the walls and shelves just look clut-

tered. In reality, the posters are from the Grateful Dead and Pink Floyd, old 1940s magazines and a photo of the Max Headroom cast and crew.

Suddenly, a horn went off and all the workers stopped. The entire building was quiet in a matter of seconds. Filming had begun on another set. Only the voice of an actor could be heard.

Live overlays

The fall season is scheduled to begin on September 18 with a special double episode, a rerun of the pilot episode titled "Blipverts" and a new episode called "Deities." Shooting began August 20.

Bruette studies the scripts before the production of each episode. In this way, he knows which graphics need to be done on each production day. After a graphic is created in advance, it is transferred to videotape and copies are sent to the Post Group and the playback editing group. The graphics are also approved by the executive producer.

In the video playback van on the set, the overlay graphic is played back on an Amiga, while the actors are performing a scene on the set. The two images are combined for the final print. Cables lead from the van to the set. The video van also performs color correction and sync on video for the show. Bruette has customized some of the equipment used here. "I have an Amiga 1000 [which] I made rack-mountable. The 2000 is 17 3/8 inches wide – just waiting to have ears put on it – so, it will be rack-mountable.

Bryce and the C-64

Actor Christopher Young plays Bryce Lynch, the computer whiz who created Max Headroom. Lynch is the director of research and development at Network 23. This role is his first for television, but he has much experience in commercials. In these advertisements, Young said he was often cast as "the wimpy kid."

Young revealed the commands he gives the computers on the set. "I type my name over and over, Christopher Tyler Young. Once in a while, I type the return key. I know a little bit about computers." Young once owned a Commodore 64, and his family had an Apple II while he was growing up in eastern Pennsylvania. In fact, Young lived very close to Commodore.

"Some of the things I say are pretty ludicrous. They don't mean a whole lot to the ordinary person, but it's fun to watch it, because the ordinary person thinks it's a bunch of incredible things that this kid is saying. In real life, most of it is just made up anyway. It is neat playing the part because it is something

continued...

different than me. I'm not like Bryce Lynch in any way, but it's fun to step out of Chris Young on character and play something totally opposite."

Did Chris study computer nerds to play this part? "No, I haven't. It came naturally. I didn't do any studying or research. I guess I took situations from my high school life, from kids I've known. Bryce is not a nerd, he's a human being. He just doesn't care what he dresses like or what he looks like, so he doesn't bother combing his hair, and he doesn't bother looking hip and trendy. He's himself, and that's what counts. He's himself, and that's why everyone likes him."

Amiga as a workhorse

Brian Frankish is the producer for Max Headroom. His jacket bears a button that says, "Who hired all these sleazy people?" and he explained that Max Headroom is a series like any other, with a fixed budget. The pilot episodes of Max Headroom went beyond the planned budget by as much as a half-million dollars each. This year, the series is a scale show with a fixed budget, meaning the crew is paid like the crew of any other show. The production staff is average sized for a series, with perhaps twenty extra people doing the computer graphics. A total of 125 people are involved in the production, including more than a dozen performers.

The filming of an episode takes seven business days, but episodes are shown every five business days, so the schedule gets tighter as the series progresses. "With six of them [episodes], you could just burn yourself out. Now, we've got to do twenty-two and last until next March."

Frankish is very enthusiastic about the use of computers in the series. "They are a necessary function in communicating the concepts of our show. We are dealing with information, where it comes from, its source and where its going. That's what these graphics are about. With a computer, we don't need a keying video switcher because the computer has a built-in keying ability. The kid [Bruette] types in the stuff, the fellows hit the right keys, it goes together in the video trailer, it plays back on the stage, whoosh; it's that simple. It is similar to what we had before. We were working with an IBM PC before. The type of PC we were using didn't have the broad range of colors or the smallness of the pixels. I don't speak 'computer-ese.' I hire guys who do. One of my major functions here is harmony."

Frankish explained that Max is something special. "We try to maintain somewhat of a camaraderie among the Max Headroom cast and crew that lets people know they are involved in a pretty phenomenal project because we are the hottest thing since sliced bread."

In the past, Frankish worked on the movies Brainstorm and King Kong, so he has worked with high technology before – and wants more technology. "The toys are getting smaller. No, come on, hey, give us more, we use it. We burn up the available technology. They say civilization hasn't grown that far yet. Come on, civilization, grow. We'll take it. If you invent something new, we'll use it. For us, the Amiga is the basic workhorse of visual communication."

•AC•

Animation for C Rookies

Part One
Quick and Dirty Bobs

By Michael Swinger

Animation on the Amiga is not really all that difficult, contrary to what some confusing and contradictory programming books would lead you to believe. Working with the system's Simple Sprites is relatively easy (although the results may be primitive, considering the Amiga's graphics potential), and there are enough sample programs that cover this kind of animation. The problem is that most of the books that deal with Amiga graphics and animation stop with Simple Sprites. The next step in complexity uses the Virtual Sprites. These are harder to program and there may be some glitches and bugs that the literature has been strangely silent about. So, there is very little guidance for the programmer whose status is less than that of a Registered-and-Baptized Developer.

Bobs (blitter objects—why aren't they called "blobs?"), the special glories of the Amiga, show the sophistication of the machine's graphics and animation to the fullest. Unfortunately, the literature is most deficient in this area, either in its silence or in its confusion. John Foust, in a column last year dealing with C programming, mentioned a yearly contest for the most convoluted and turgid C program. I would like to nominate the sample program that appears in the Rom Kernal Manual at the end of the graphics section as one of the Official Amiga Developer entries. If that kind of spaghetti code were written in BASIC, everyone would scoff!

This series of articles is written by a beginning C programmer for other beginners who want to explore graphics and animation in the simplest, most direct way. These suggestions may not always be "K&R approved", but they work. They are the result of long, frustrating hours spent trying to find where all the bodies are buried. I hope they will provide a skeleton for more elaborate and adventuresome programming on your part.

The first program simply opens a screen and window in 320 x 200 with 32 colors, places a small bob on screen, and waits for you to click the window closed. When you try to display more than one bob, some screen flicker occurs, so, in a future program, we will introduce double-buffering. A third program will cover the machine routines for handling animation objects ("AnimObs" and "AnimComps"). We will discuss some of the crucial information about animation objects omitted from the Rom Kernal Manual.

You can create the images for the bobs as Deluxe Paint brushes. The brush files will then have to be converted from IFF format to a form that the machine actually understands. For this translation, you will need a public domain utility program named "gi" (available on the Fish disks). Unfortunately, gi will work only with the original Deluxe Paint. A DPaint II file contains extra color-cycling information, so you will have to save your brushes and edit them in gi with the old version of Deluxe Paint (Aegis Images "windows" should work with gi, but I haven't tested them).

There are several other editors in the public domain (IFFDump is one), but a gi file requires minimal re-editing to prepare a file for your program. Gi will also produce color information, but it will include only those colors actually used in your brush. If your brush doesn't use all 32 colors, you might want to make a special brush that does include all the colors. There is one small error in a gi file (an extra comment mark—*/), so look at the sample program below for one way to edit your gi file.

I am using the Aztec compiler, so I specify the +L option and the 32-bit library (Casting is still something of a mystery to me, so I'm taking the easy way out right now.). If you are using Lattice, change the .h files as necessary. If you have expanded your machine's memory beyond 512K, or if you anticipate that your program will be run on an expanded machine (a good possibility), you will also want to include any compiler options that will place the data in Chip memory.

We will also be programming in the Intuition environment. Working with Intuition requires more memory and machine overhead than working directly with some of the Kernal primitives, but Intuition does a lot of the dirty work in allocating and deallocating memory and dealing with the mouse and its messages — and besides, it does windows .

continued...

```
/* Program One-a simple bob. The */
/* notes precede the relevant    */
/* statements and are explained  */
/* after the program listing.    */

#include <functions.h> /* Manx only */
#include <intuition/intuition.h>
#include <graphics/gels.h>
struct IntuitionBase *IntuitionBase;
struct GfxBase *GfxBase;
struct Screen *Screen1;
struct Window *Window1;
struct ViewPort *WVP1;
struct VSprite s1, s2;
struct GelsInfo gelsinfo;
/** NOTE 1 **/
struct collTable Ctable;
#de ne RP1 Window1->RPort
VOID Drawit();
/** NOTE 2 **/
USHORT colormap[32] = {
0x0ccd,0x0585,0x0f79,0x0f30,0x0f9b,
0x0f90,0x0fc3,0x0fc9,0x0eef,0x0ddd,
0x0ccc,0x0aaa,0x0fdc,0x0fcb,0x0fba,
0x0ea9,0x0e93,0x0d90,0x0c87,0x0a54,
0x0c75,0x0f67,0x0555,0x0069,0x09e1,
0x0fed,0x07c1,0x05a0,0x0270,0x0fdd,
0x0fff,0x0000 };

WORD Image_data1[130] = {
/* Width: 18 (pixels)
Height: 13 (pixels)
Depth: 5 (planes)*/
0x0000,0x0000,0x0000,0x0000,0x0080,
0x0000,0x0080,0x0000,0x0080,0x0000,
0x0080,0x0000,0x0080,0x0000,0x0080,
0x0000,0x0080,0x0000,0x0080,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,
0x0000,0xffff,0x8000,0xffff,0x8000,
0xfc1f,0x8000,0xfc1f,0x8000,0xfc1f,
0x8000,0xfc1f,0x8000,0xfc1f,0x8000,
0xfc1f,0x8000,0xfc1f,0x8000,0xfc1f,
0x8000,0xffff,0x8000,0xffff,0x8000,
0xffff,0x8000,0x0000,0x0000,0x0000,
0x0000,0x03e0,0x0000,0x03e0,0x0000,
0x03e0,0x0000,0x03e0,0x0000,0x03e0,
0x0000,0x03e0,0x0000,0x03e0,0x0000,
0x03e0,0x0000,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0360,0x0000,0x0360,
0x0000,0x0360,0x0000,0x0360,0x0000,
0x0360,0x0000,0x0360,0x0000,0x0360,
0x0000,0x0360,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000
};

/** NOTE 3 **/
WORD Sbuffer1[2 * 13* 5];
WORD Cmask1[2 * 13];
WORD Bline1[2];

struct NewScreen NewScreen1 ={
0,0,320,200,5,1,0,
NULL,CUSTOMSCREEN,
NULL,NULL,NULL,NULL };

struct NewWindow NewWindow1 ={
0,0,320,200,1,0,CLOSEWINDOW,
SMART_REFRESH | ACTIVATE |
BORDERLESS | WINDOWCLOSE,
NULL,NULL,NULL,NULL,NULL,0,0,0,
0,CUSTOMSCREEN };

struct VSprite v1 ={
NULL,NULL,NULL,NULL,NULL,NULL,
OVERLAY | SAVEBACK,
0,0,13,2,5,0,0,&Image_data1[0],
&Bline1[0],&Cmask1[0],NULL,NULL,
0x01f,0,NULL };

struct Bob b1 ={
NULL,&Sbuffer1[0],&Cmask1[0],NULL,
NULL,&v1,NULL,NULL,NULL };


main()
{
Open_Libraries();

Open_Screens();

Init_Bobs();

Drawit();

Cleanup();

} /*** end main ***/

/** NOTE 4 **/
Open_Libraries()
{
IntuitionBase = (struct
IntuitionBase *)
OpenLibrary("intuition.library",0);

GfxBase = (struct GfxBase *)
OpenLibrary("graphics.library",0);
return();
}

Open_Screens()
{
Screen1=OpenScreen(&NewScreen1);
NewWindow1.Screen=Screen1;
Window1=OpenWindow(&NewWindow1);
WVP1 = (struct ViewPort*)
ViewPortAddress(Window1);
LoadRGB4(WVP1,&colormap,32);
return();
}

Init_Bobs()
{
gelsinfo.nextLine = NULL;
gelsinfo.lastColor = NULL;
gelsinfo.collHandler =NULL;
RP1->GelsInfo = &gelsinfo;
v1.VSBob=&b1;
InitGels(&s1, &s2, &gelsinfo);
InitMasks(&v1);
v1.X=25;
v1.Y=75;
AddBob(&b1,RP1);
/** NOTE 5 **/
/*RemBob(&b1);*/
return();
}
Cleanup()
{
Wait(1<<Window1->UserPort->mp_SigBit);
CloseWindow(Window1);
CloseScreen(Screen1);
CloseLibrary(GfxBase);
CloseLibrary(IntuitionBase);
return();
}

/** NOTE 6 **/
VOID Drawit()
{ SortGList(RP1);
WaitTOF();
DrawGList(RP1,WVP1);
}
```

NOTE 1

We are not going to be concerned with collision checking, but the program will crash if this structure is not included.

NOTE 2

The array for your colormap is produced by gi. It will be the same for all bobs in your program, so you can save just one file as a separate colormap and edit it from the rest of your bob data files. The bob data itself can be edited as shown. Be sure to remove the extra comment mark that gi writes into the file. The VSprite structure requests the image data as a WORD, but gi marks it as UWORD. Change it if you get tired of compiler warnings.

NOTE 3

If you have only one bob on the screen at once and the bobs will never overlap, or if memory is tight, you can eliminate the CMask and BLine. If you eliminate these, you must also eliminate the call to InitMasks(). If all your bobs are exactly the same size and will be shown at exactly the same screen coordinates, or if there is nothing in the background that needs to be restored, you can also save much memory by eliminating the SBuffer! If you eliminate any of these buffers, specify NULL in the appropriate places in the VSprite and Bob structures. Note that the width of the bob is always specified as the WORD width — the pixel width, divided by 16.

NOTE 4

These calls do not use error checking because it seems superfluous (I know, I know...).

NOTE 5

If you want to remove a bob from the display list, there are two statements you can use. RemIBob(&b1,RP1,WVP1) removes the bob immediately when the statement is called. RemBob(&b1) does not remove the bob from the Gel list until SortGList and DrawGList are called. If you are doing your own sequenced animation by alternating between several versions of a bob, the first call (RemIBob) can cause some flicker. Note the syntax for the second

call (RemBob) — it is defined as a Macro in the gels.h file and the syntax is shown incorrectly in a number of books.

NOTE 6
The Drawit() function must be called each time you do something to the Gel list such as, adding or removing bobs or changing their screen coordinates. This function is defined as a VOID because we don't care whether or not it returns anything to the calling function.

In a future installment, we will discuss double buffering, which is unfortunately necessary if you want to have multiple bobs on screen. You might be able to get by without it, if your bobs are small and use fewer than 32 colors. If you choose this method, the bobs look like they were done on a C=64, not an Amiga . . . so, you may as well use the sprites.

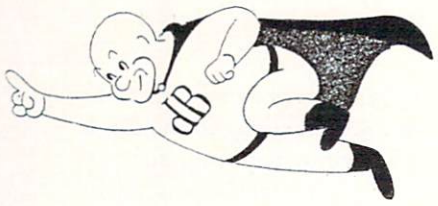
You can try to plow through the Rom Kernal Manual, but there are better and more accurate information sources available for further reading.

Inside the Amiga by John Berry (Sams Books) is a good, very readable tutorial in C, but the graphics stop with Simple Sprites. Robert Peck, who wrote the graphics section in the RKM as part of the original Amiga team, has written the Programmer's Guide to the Amiga (Sybex Books). This publication makes a lot more sense. Inside Amiga Graphics by Sheldon Leemon (Compute Books) is a good introduction — I owe a great deal to this book — but it needs to be updated to reflect changes in the 1.1 and 1.2 versions of the operating system. None of these books explains the built-in animation routines, but an article by Roy Thompson has just appeared in Ami Project, Volume 1 #7, which finally clears up much of the mystery. Read that article for a preview of our next installment.

•AC•

dBMAN

Relational Database Management System Version 3.00A



reviewed by Clifford Kent

In the (microcomputer) beginning there was the 8080 (and Z80) and it ran CP/M. As soon as floppy disk storage came about, these primitive microcomputers were able to do database management. The next step was the birth of dBASE, soon to grow into dBASE II, a real business tool. Some might say that CP/M, Word Star and dBASE II were enough to make the early microcomputer a business machine.

Soon, the 8088 (and 8086) and MS-DOS were born (as a clone of CP/M, but that's another story). dBASE II was soon available for the new business standard microcomputer. Later, dBASE III and dBASE III+ improved on a good thing and made it possible to do REALLY SERIOUS database applications on a microcomputer. As with all big software hits, there were many dBASE imitations- - - some might even be better than the original.

dBMAN was first available for the PC-DOS/MS-DOS world. Unlike the name brand product, however, dBMAN is available for the Amiga. The original Amiga release, dBMAN 2.02, mixed some features from dBASE II and dBASE III. It was a mature product from the start (meaning it didn't crash without operator help and it never lost data). We've used dBMAN 2.02 in our business for the last eight months to handle small (under 1000 record) databases and it has performed well.

The newest release is dBMAN Version 3.00. The new version, which resembles dBASE III, has done well in most of our tests. It is more like dBASE III. Its improved user interface makes it a much better product for the Amiga.

For those who have never heard of any of the names dropped above, I should explain:

Database - a computer program for keeping and using records, ranging from simple filing programs (the computer equivalent of 3x5 cards) to artifically intelligent wonders that won't even fit in my 2.5 megabyte Amiga.

Relational Database - a useful, sophisticated program that can store information in more than one computer file, then reconnect ("relate") multiple data files in more than one way, as the human operator's imagination requires.

dBASE III - the best known and most succesful relational database program on PC Compatibles. The size and complexity of the data is limited mostly by the available disk space. The built-in programming language lets you create highly specialized and very sophisticated data handling programs.

Relational databases are fairly easy to set up and surprisingly flexible for extracting information from the stored database.

My experience with microcomputer databases includes dBASE II on CP/M computers, dBASE III and FoxBase+ (perhaps the best of the dBASE copies) on PCs and dBMAN 2.02/3.00 on the Amiga. I've set up several full scale business record keeping systems, as well as customer databases, simple mailing lists, and even my personal checking with these software tools. They all work ... there are many differences, but they can all do the job.

dBMAN On The Amiga

In this article, I describe dBMAN 3.00 in detail and compare it to its relatives running on a PC compatible machine (including the SideCar and Amiga 2000 Bridge Card). My aim is to present a clear picture of what can be done with dBMAN 3.00 and how quickly it can happen.

The dBMAN package includes an 8" x 9" three ring binder, containing just over an inch of manual and a 70 percent full single 880k disk. The disk is not copy protected and does not include Amiga Workbench. VersaSoft recommends a hard disk for efficient use, but dBMAN can run surprisingly well with two floppies and 256k or more of FAST RAM. Nearly half of the manual describes dBMAN and its basic function. The remainder of the manual is a command and function reference section that becomes very important as you gain experience.

There are 28 files on the dBMAN disk, but only five would be included on a normal work disk, leaving just over 512k free on an Amiga floppy to hold data files and dBMAN programs. If your database will be larger than 400k, a hard disk will probably be needed. The remaining 23 files on the dBMAN disk contain updates to the manual, two text files on converting existing dBASE II and dBASE III applications to dBMAN and four application programs written in the dBMAN command language.

continued...

NOW SHIPPING

Special

Introductry Price

Complete Communications Package

300/1200 1 Year warranty

300/1200 Fully Hayes compatable Modem - 2 Year warranty

$129.00

(Modem, Cable & Software)

300/1200/2400 Fully Hayes compatible modem CCITT - 2 Year warranty

$249.00

(Modem, Cable & Software)

Call or write for information about our other great products for the Amiga or our Demo disk $5.00

The REASON system is a series of programs designed to aid writers and editors in editing documents.

REASON programs do three things:

*proofread input text *analyze the style of input text *provide help about English usage

Many options give editorial comments and suggestions.

The **REASON** system finds potential errors, then you decide which potential errors need correcting. Thoughtful use of the **REASON** system can help both the experienced and inexperienced writer.

With the REASON **system, there are six main options:**

1. **Prose** describes the writing style of a document, namely, readability and sentence characteristics, and suggests improvements. **Prose** compares a document with standards for one of several document types. INSTRUCTIONAL TEXT will compare input text with good training documents. TECHNICAL MEMORANDA will compare input text with good technical memoranda. And USE CUSTOM STANDARDS will compare input text with any user created standard.
2. **Style** finds sentences that contain passive verbs, expletives, noun nominalizations, and multiple nominalizations. Also, Style will give a readability level for each sentence in the input text or find sentences that are equal to or greater than a specifically defined readability level. Another function performed by style is to find sentences that have a specifically defined length (number of words contained in a sentence).
3. **Word Analysis** will check the input text for general diction, sexist terms, sentences that contain forms of the verb "to be", acronyms and abstract words.
4. **General Structure** checks input text for general organization, general topics, sentence breakdown (parts of speech) and syllable breakdown (syllable count of each word)
5. **Proofread Document** checks for possible spelling errors, double words, possible punctuation errors, diction and split infinitives.
6. **Extra** allows access to AMIGA Preferences and Build Custom Prose Standard.

Requires an AMIGA computer with 512K

AMIGA is a Registered Trademark of Commodore Amiga

$395.00

THE OTHER GUYS

55 North Main Street
Suite 301-D
PO Box H
Logan Utah 84321

(801) 753-7620
(800) 942-9402

VISA MasterCard

dBMAN has no Workbench icon and should be invoked with the CLI commands: STACK 9000 RUN DBMAN <Program-Name> These commands can be placed in an EXECUTE file for simplicity. Also, public domain programs are available to invoke EXECUTE files from Workbench Icons, if you prefer the Workbench interface. The <Program-Name>, used to automatically start an application program written in the dBMAN command language, is optional.

dBMAN creates its own screen with no window borders and only two colors. A custom screen, needed on the Amiga to get an 80x24 display, is required to run dBMAN (or dBASE) programs written originally for other systems. Although the dBMAN screen does not have a title bar with front/back gadgets, you can still multi-task dBMAN by using the <Left-Amiga-A>+<N> keys to get to the Workbench Screen and the <Left-Amiga-A>+<M> keys to return to dBMAN. The two color display was probably chosen to save memory (more bit planes use more RAM), but it can sometimes cause confusion. Reading the screen can be tricky because the Amiga's cursor doesn't blink and dBMAN makes extensive use of reverse video.

If dBMAN is started without a program name on the command line, you get a split screen with two lines for command entry, two lines for help and error messages and 18 lines of data display space. You have a command interpreter with hundreds of commands and functions for creating, editing, sorting, indexing, selecting, finding, listing, printing, averaging, totalling, counting and reporting.

If you know dBASE, you'll feel right at home. Commands are available to import existing dBASE II and dBASE III databases (There are no export commands, but that's also easy to do using an intermediate text file). The commands and functions are not all identical to dBASE, but they are very similar and the differences are well documented. If you have written a dBASE II or dBASE III application, you should have very little trouble moving both existing data and programs to dBMAN. If you have an existing dBASE application written by someone else, porting it to dBMAN provides an excellent way of learning the language. If you need help, there are many books vailable on the use of dBASE III.

The only thing dBMAN won't support is the dBASE III "memo" data type (a free form - word processor file that is logically attached to the regular data base). The most obvious use I've found for memo data in dBASE III is storage for examination notes in a doctor's patient database. If the doctor uses memo data, his notes are easy to find. If he doesn't use it, only 10 bytes of disk space is wasted per patient. Note keeping could be handled with a dBMAN program, but the dBASE solution is much more elegant.

If you are a Basic programmer, you will find the dBMAN language easy to learn. A basic set of tools for structured programming are available: PROCEDURE calls IF...ELSE...THEN DO WHILE...LOOP...EXIT...ENDDO DO CASE...OTHERWISE...ENDCASE. You can store information in memory variables, as well as in the database files. Variables can be either global or local in scope to a single PROCEDURE. A full set of relational operators, arithmetic operators, string functions, date functions, data validation, and data formatting and display commands are also included. Menus can be created for keyboard or mouse control of a program. DBMAN language is a real programming language, not a glorified macro-keystroke recorder. In my opinion, it is a much better language for business programming than Basic, Pascal, Forth or C.

If you want to run a dBMAN application program, include its name in the dBMAN command line. In this case, the opening dBMAN screen is replaced by the application program's opening screen and dBMAN is almost invisible, as the skill and sophistication of the application programmer shines through.

Application programs are stored in standard Amiga text files that are interpreted at run time. This point creates a very nice programming environment on the Amiga. I RUN dBMAN and then RUN a copy of TxEd of each program module I want to work on. During development, each procedure is stored in its own file. After the program is complete, all the procedures can be assembled into a single file for execution.

dBMAN Programs

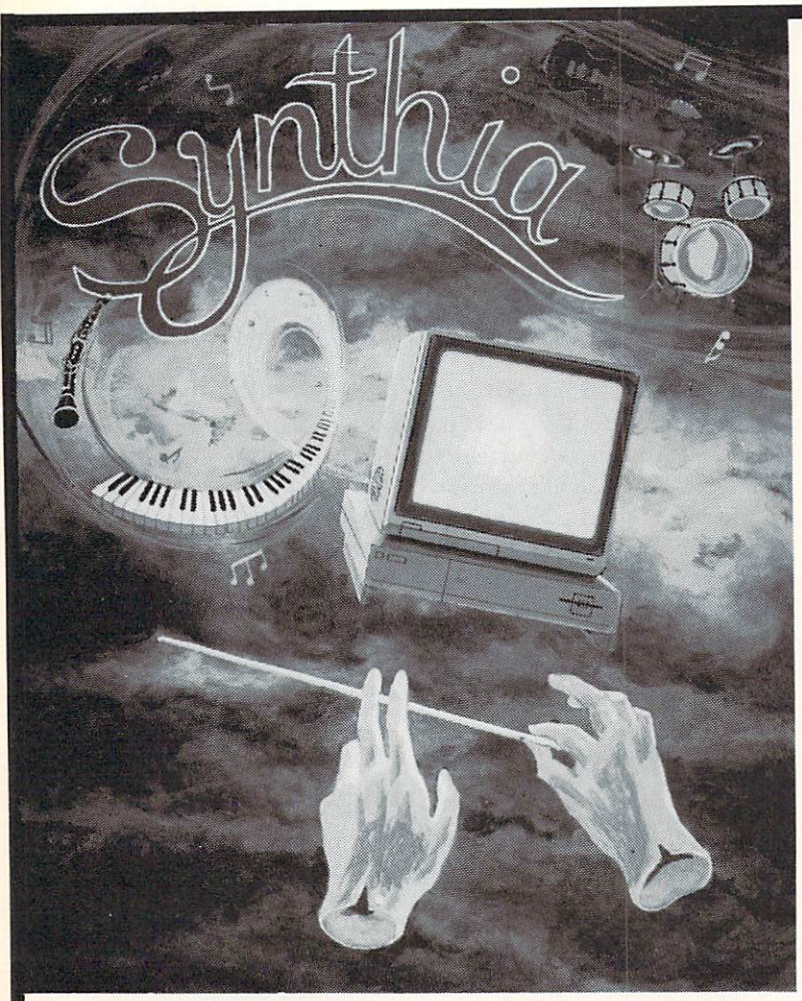
Four application programs are supplied with dBMAN. In addition to being directly useful, each is a good example of dBMAN programming.

The CLI command "RUN dBMAN TUTOR" starts an application program designed to teach basic dBMAN to the beginner. The program is menu-driven and a good starting point for learning dBMAN from scratch, leading you, step by step, through creating and using a database.

The CLI command "RUN dBMAN ASSIST" starts an application program that replaces the standard command line control with a mouse operated pull-down menu interface. This application is an excellent example of dBMAN language in action. dBMAN is transformed into an "Amiga Program" and you have the source code, so you can customize the user interface to your liking. The possibilities are terrific. I expect to see a new generation of easy-to-use, mouse controlled business applications that are also reliable and sophisticated because they are based on a mature database system and years of program development.

ASSIST has eight pull-down menus with the basic database functions logically grouped. Each selection leads you through the selected operation, offering choices as needed. After

continued...



Other Products From The Other Guys	
Reason	$395.00
Omega File	$79.99
Promise	$49.99
KEEP-Trak GL	$49.99
AMT (Amortization Program)	$39.99
Match-It	$39.99
Math-A-Magician	$39.99
Talking Story Book (Christmas Stories)	$39.99
Musicial Slide Show Demo	$ 5.00

Call or write for more information.

SYNTHIA *High Performance Digital Synthesizer*

A state of the art music tool which will:

Create digital **IFF Instruments** for use with nearly all music programs!

Modifying existing **IFF Instruments.** Use *SYNTHIA* on digitized samples to add reverb, wow, and other enhancements.

SOMETHING FOR EVERYONE:

Additive Synthesis - a traditional method which can create almost any type of instrument.

Plucked String Synthesis - simulates plucked strings . . . right down to the 'pluck'.

Interpolative Synthesis - a method which introduces the natural imperfections found in instruments. (Instruments such as brass, woodwinds, pianos, etc.)

Percussion - build your own drum set . . . create any drum you desire.

Subtractive Synthesis - a simple method of creating instruments.

Special Effects - includes filtering, amplification, phasing, waveshaping, amplitude modulation, real reverb, and . . .

IFF Music Player - powerful and compact. Now you can enjoy those songs that needed a memory expansion before! Up to 32 tracks and 32 IFF Instruments! Supports chords, ties, etc.

NOW SHIPPING

IS IT LIVE . . . OR IS IT SYNTHIA?

Synthia uses the latest technology to generate realistic sounding instruments and even the new families of instruments sound real. A real synthesizer on a real computer!

Why buy digitized instruments when you can *SYNTHIAsize* them?

Requires AMIGA 512K

$99.99

THE OTHER GUYS

55 North Main Street
Suite 301-D
PO Box H
Logan Utah 84321

(801) 753-7620
(800) 942-9402

VISA MasterCard

you've made your menu choices and typed in the needed file names, search strings, etc., ASSIST displays the resulting dBMAN command line before it is executed. With this information, you can learn about the dBMAN language while you are using it.

When ASSIST is running, the dBMAN title bar and front/back gadgets are available, so you can control the dBMAN screen with the mouse to do something on the Workbench screen.

Most database users would be satisfied with dBMAN and ASSIST as their everyday database. The availability of the source code and a complete database programming language become a great help as the user's understanding and needs expand.

Sample dBMAN Program

I've included the source code for a simple, public domain mailing list handler. I initially wrote the program for dBASE II. I then ported it to dBMAN 2.02 on our first Amiga and finally to dBMAN 3.00 (in only a few hours). The program does not fully exploit the capabilities of dBMAN 3.00, but it may serve as interesting reading if you'd like to look at a simple application.

MailList opens with an on-screen menu listing all the program's functions. I won't try to describe the functions here, but a few points will help explain dBMAN programming. The main program is a "DO WHILE" loop that displays the menu, gets input from the keyboard and executes the selected PROCEDURE in a big CASE statement. dBMAN/dBASE programs, unlike Pascal programs, have the main program at the beginning of the listing with PROCEDURES following.

MailList maintains its own variables for the path name and the data file name. Having both names maintains the flexibility to place the data in RAM: when I want speed, and can save typing by saving things.

In the Open-File PROCEDURE, you see the line:

```
USE &Dname&Fname
```

The ampersand ("&") is the macro symbol. This signal tells dBMAN that the current contents of a memory variable should be included in a program command. In this case, the variables "Dname" and "Fname" are used to complete the command line when the code is executed. In this simple example, MailList allows the path to be saved in one string variable and the file name to be in another. When it's time for the "USE" command to open the data file, the command verb and the two strings are assembled into a complete command. This use of macro substitution started with dBASE II, and continues as a handy short-cut and the programmer's last resort when he wants dBMAN to do something unusual.

Screen formatting is done using the "@ <row>,<col>" command sequence to move the cursor to the right position on the screen (or printer). The "SAY" command shows a message. The "GET" command displays the current contents of a data field or a memory variable and makes it available for editing. Normally, a series of SAY and GET commands ends with the READ command, where the actual data entry takes place.

Both the Set-Filter and Set-Index PROCEDUREs in MailList assume some understanding of dBMAN's command syntax. Fairly complex data control is possible, but the user must know how to phrase the command. Much more friendly systems have been designed (dBMAN's ASSIST program is a good example), but this program was written for in-house use and friendliness wasn't worth the trouble at the time.

The MailList code includes many lines ending with semicolons. This syntax allows the programmer to enter a single line of command code (up to 236 characters) spanning several physical lines in the source code file.

In the two hard copy routines, Print and Labels, you will notice something very unusual for an Amiga program - printer escape sequences. dBMAN uses the PAR: device, not PRT:, so the Amiga printer drivers are not used. If you want something special from your printer, you'll have to look up the printer codes. Use of printer codes is the norm for other computers and must be available to run applications written for other computers. Unfortunately, I can't find a way to use the AmigaDOS printer drivers for the simple things other computers handle automatically.

In a future issue of *Amazing Computing*, MailList will be "Amigatized" and extended into a more complete and easier to use application program. For now, I hope the sample code introduces dBMAN as a programmer's tool.

Benchmarks

Table 1 includes the results of three very simple tests of database speed.

Two computers were used: an Amiga 1000 and a PC-XT clone. The Amiga 1000 was equipped with 3 floppy disks and 2 Megabytes of zero wait state fast ram. No hard disks were available at the time, but the tests were run using both floppy disk and RAM disk storage. Hard disk performance should be intermediate. For the floppy disk tests, the AmigaDOS command 'AddBuffers' assigned 30k of cache ram to each disk drive. This allotment improves performance for many tasks (including database reads), but does little to help the speed of disk writes.

The second computer was a PC-XT clone with 2 floppy disks, 2 hard disks, 640k ram, NEC V20 processor and an 8087 math processor. Although the PC runs at 8 MHz, the tests were run at the standard 4.77 MHz clock speed.This adjustment should make the PC's performance similar to an Amiga Side Car or an Amiga 2000 with Bridge Board.

Three database programs were tested: dBMAN 3.00, dBASE III and FoxBase+. The first two programs shoud be familiar. The third program, FoxBase+, is a dBASE compatible compiler. Fox is included here because it represents the current state of the art system for PC compatibles.

The first test involved indexing a mailing list of 593 entries by Zip Code. Indexing is similar to sorting but, rather than physically rearrange the data records, the program creates a second file containing the information necessary to locate the database records in the correct order.

The second test used an index file built on last names to FIND the 590th out of 593 entries. As you can see from the test results, data access is very fast if the correct index exists. Many programmers will maintain a collection of index files for the same database to minimize data search times.

The third test used LOCATE, a sequential search of the database, to find the 590th out of 593 entries. In this case, the database was not indexed. I expected this test to be a simple look at sequential disk read speed, but the times suggest that more is involved here.

From these simple tests and lots of hands on use, I can safely draw some conclusions.

First, never use data AND index files on a floppy disk. Unless the data file is very small, the grinding of the drive will drive you crazy. If necessary, copy the index files to RAM: at the start of your program, then back to floppy at the end.

dBMAN on an Amiga will generally out-perform dBASE III on a plain PC. This difference should not be a big surprise- - - most of us already know that an Amiga is faster than a PC. The interesting point is that dBMAN performs very well using floppy disk storage if the program also uses the RAM: Disk in the right places.

Be careful not to read too much into the floppy/ram timings. The floppy slow considerablys when reading a file whose physical order is very different from the indexed order. For example, if the file is sorted by last name, then indexed by city, Listing the filewill require many disk seeks. Hard disks seek faster than floppy disks. RAM disk seek is best of all.

dBMAN Run-Time Utilities

In addition to the standard dBMAN interpreter, a dBMAN Run-Time Utility Package is also on the market. This package "tokenizes" the completed application program text file, creating a version of the program that cannot be read or changed with normal computer tools. The Run-Time code executor runs this encrypted version of the program. The Run-Time Uttilities give the program developer three advantages:

•Your source code is secure.

•Your client does not need to buy dBMAN. The distribution licence for the Run-Time executor is included in the price of the package.

•Your code runs noticeably faster because the original text file has been converted to a semi-compiled code form that can be processed faster at runtime.

I did not properly benchmark dBMAN Run-Time's performance- - - that process would require some careful programming. However, I did write a simple program that made a lot of PROCEDURE calls and did some string counting and arithmetic while reading through a database. Run-Time executed the test program about 20% faster than the dBMAN interpreter.

I also used Run-Time to compile several application programs and noted a significant speed-up in many operations. In particular, the dBMAN ASSIST program had a quicker feel. It appears that complicated dBMAN programs benefit more than the simple commands, like INDEX or LOCATE.

Conclusions

VersaSoft is an established supplier of database software for a wide variety of computers. dBMAN 3.00 is quite new for the Amiga and a few problems still exist. I have discussed the problems with VersaSoft's knowledgeable, candid tech support staff. dBMAN 2.02 was very reliable and expect to see a bug-free dBMAN 3.00 soon.

Who should use dBMAN?

Anyone who wants to run an existing dBASE/dBMAN application on an Amiga should use dBMAN. Dozens of well designed public domain programs in the CP/M and MS-DOS libraries can be moved to the Amiga easily.

If you need an efficient Amiga programming language for business or record keeping applications, you should also consider dBMAN. dBMAN includes a very high level language for entering, storing, retrieving and reporting information. It can be used to quickly create convenient,

continued...

reliable programs in days. The same functionality could undoubtedly be written in C or Basic, but development would take much longer.

dBMAN is also out there for any of the thousands of dBASE programmers looking for a way to widen their market without learning a whole new language.

About the Author

Cliff Kent is a full-time programmer and Amiga developer. His first Amiga project was a Forth compiler for the Amiga used to create MacroModem, a telecommunications program. Kent is currently working on a business software package aimed at PC compatibles. Anyone interested in dBMAN/dBASE programming on the Amiga should send Email to PeopleLink ID c.kent or CompuServe ID 72437,162.

File Structure For MailList

Name	Type	Size
LAST	Character	30
FIRST	Character	15
STREET	Character	25
CITY	Character	17
STATE	Character	2
ZIP	Character	5
PHONE	Character	13
NOTES	Character	15
		122

```
SET TALK OFF
*
* MailList.cmd
* Mailing Labels system main menu program
*
* Last dBASE II version 27Apr84
* Converted for dBMAN 01Nov86,02NOV86,03Nov86
* Converted for dBMAN Run-Time 03Jul87
*
SET PROCEDURE TO MailList
CLEAR
SET CONSOLE ON
SET PRINT OFF
SET DELETED ON
Dname = ' '
Fname = ' '
Key = ' '
FilterFld = ' '
FilterStr = ' '
FilterType = ' '
IndexStr = ' '
Delim = ','
M1  = 'Open disk file,'
M2  = 'Filter data,'
M3  = 'Index data,'
M4  = 'Add new names,'
M5  = 'Edit names,'
M6  = 'List file to video,'
M7  = 'Browse file on video,'
M8  = 'Print file master,'
M9  = 'Print labels,'
M10 = 'Save file,'
```

continued...

dBASE Compatible Products

dBMAN 3.00 Interpreter $199.95
Run-Time Utility Package..... $199.95
Interpreter & Run-Time..... $295.00

dBMAN is also available for IBM compatibles, IBM compatibles with Local Area Networks, Xenix, Atari ST and Macintosh

VersaSoft Corporation
4340 Almaden Expressway, Suite 250
San Jose, CA 95118

---

dBASE III Plus $695.00

Ashton-Tate
20101 Hamilton Ave.
Torrance, CA 90502-1319

FoxBase+ Development Package $395.00
Royalty-Free Runtime..... $500.00

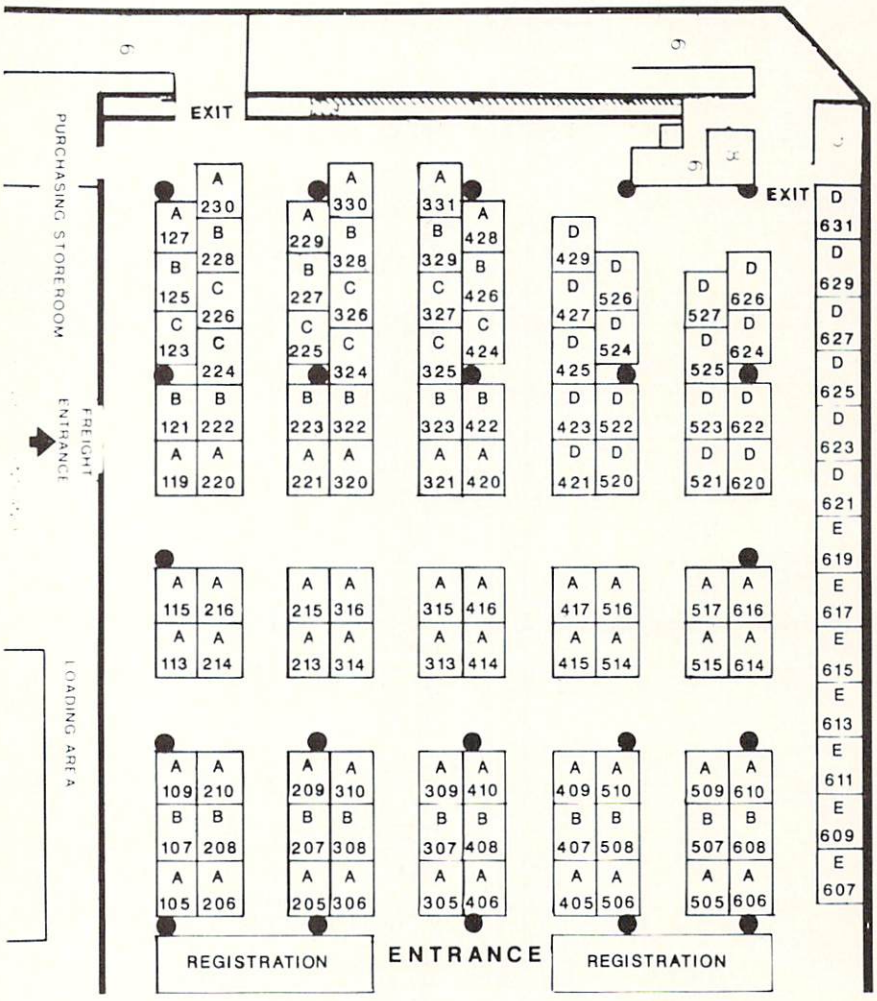
Fox Software
27493 Holiday Lane
Perrysburg, OH 43551

Table1 *Simple Benchmarks*

	Index 593 records on ZIP Code:	
dBMAN 3.00	Data RAM: Index RAM:	25.6 seconds
	Data DF0: Index RAM:	35.3 seconds
	Data DF0: Index DF0:	278.4 seconds
dBASE III	Data & Index on hard disk	53.4 seconds
	Data & Index on floppy	94.6 seconds
FoxBase+	Data & Index on hard disk	7.6 seconds
	Data & Index on floppy	13.0 seconds
	Find record 590 out of 593 using index:	
dBMAN 3.00	Data RAM: Index RAM:	.3 seconds
	Data DF0: Index RAM:	1.1 seconds
	Data DF0: Index DF0:	2.2 seconds
dBASE III	Data & Index on hard disk	.3 seconds
	Data & Index on floppy	2.1 seconds
FoxBase+	Data & Index on hard disk	.3 seconds
	Data & Index on floppy	.8 seconds
	Locate record 590 out of 593 with text search:	
dBMAN 3.00	Data on RAM:	11.8 seconds
	Data on DF0:	21.6 seconds
dBASE III	Data on hard disk	7.5 seconds
	Data on floppy	32.2 seconds
FoxBase+	Data on hard disk	3.6 seconds
	Data on floppy	8.5 seconds

Ami Expo

Comes to Los Angeles!



January 16–18, 1988
The Westin Bonaventure
Los Angeles, California

For Exhibitor or Attendee Information
Call 800–32–AMIGA Nationwide
In New York, 212–867–4663

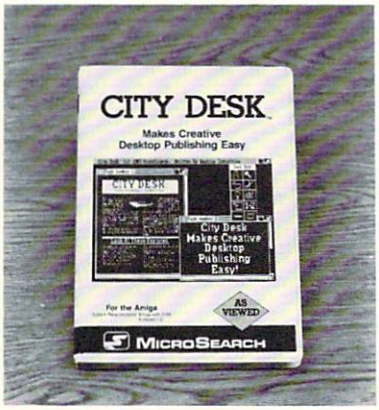
Amiga(TM) is a registered trademark of Commodore–Amiga, Inc.

```
M11 = 'dBMAN commamd mode,'
M12 = 'Quit'
MenuText = M1+M2+M3+M4+M5+M6+M7+M8+M9+M10+M11+M12
RELEASE M1,M2,M3,M4,M5,M6,M7,M8,M9,M10,M11,M12

DO WHILE Key <> 'Q'
  ERASE
  @ 0,36 SAY 'MailList'
  @ 1,28 SAY '-----------'
  @ 2,32 SAY 'By Clifford Kent'
  @ 4,30 SAY 'File    - '+FILENAME()
  @ 5,30 SAY 'Filter - '+FilterFld+" "+FilterType+" ";
           +FilterStr
  @ 6,30 SAY 'Index   - '+IndexStr

  ASSIGN VMENU(Delim,MenuText,Row(),1,2,1,-1)

  ERASE
  DO CASE
    CASE VMENU()=1
      DO Open-File
    CASE VMENU()=2
      IF FILENAME() <> ''
        DO Set-Filter
      ENDIF
    CASE VMENU()=3
      IF FILENAME() <> ''
        DO Set-Index
      ENDIF
    CASE VMENU()=4
      IF FILENAME() <> ''
        DO Add
      ENDIF
    CASE VMENU()=5
      IF FILENAME() <> ''
        DO Edit
      ENDIF
    CASE VMENU()=6
      IF FILENAME() <> ''
        DO List
      ENDIF
    CASE VMENU()=7
      IF FILENAME() <> ''
        BROWSE FIELDS Last,First,Phone,Street,City,State;
                  ,ZIP,Notes
      ENDIF
    CASE VMENU()=8
      IF FILENAME() <> ''
        DO Print
      ENDIF
    CASE VMENU()=9
      IF FILENAME() <> ''
        DO Labels
      ENDIF
    CASE VMENU()=10
      IF FILENAME() <> ''
        USE &Dname&Fname
        DO Re-Order
      ENDIF
    CASE VMENU()=11
      ? 'The current file and index are still open.'
      ? 'Type "DO MENU <CR>" to return to menu.'
      RETURN
    CASE VMENU()=12
      Key = 'Q'
  ENDCASE
ENDDO
QUIT


* Select drawer and open file.

PROCEDURE Open-File
ERASE
IF FILENAME() <> ''
  USE
ENDIF
Dname = Dname+DUPCHAR(RANK(' '),30)
Fname = Fname+DUPCHAR(RANK(' '),26)
@ 5,13 SAY 'Drive and/or directory name: ';
       GET Dname PICTURE DUPCHAR(RANK('!'),30)
@ 7,5  SAY 'Data file name (.DBF will be added): ';
       GET Fname PICTURE DUPCHAR(RANK('!'),26)
SET CONFIRM ON
READ
Dname = TRIM(Dname)
IF SUBSTR(Dname,LEN(Dname),1) <> ":";
   .AND. LEN(Dname) > 0;
   .AND. SUBSTR(Dname,LEN(Dname),1) <> "/"
   Dname = Dname + "/"
ENDIF
Fname = TRIM(Fname)
@ 9,5 SAY 'Searching for '+Dname+Fname+'.DBF'
IF FILE(Dname+Fname+'.DBF')
  USE &Dname&Fname
  DO Re-Order
ELSE
  @ 11,5 SAY 'File not found - press <RETURN> '
  WAIT
ENDIF
RETURN


* re-filter and re-index

PROCEDURE Re-Order
IF TRIM(FilterFld)<>'' .AND. TRIM(FilterType)<>'';
                       .AND. TRIM(FilterStr)<>''
  Key = FilterFld+" "+FilterType+" "+"'"+FilterStr+"'"
  @ Row()+2,5 SAY 'Filtering file.'
  SET FILTER TO &Key
ELSE
  SET FILTER TO
  FilterFld = ' '
  FilterStr = ' '
  FilterType = ' '
ENDIF
IF TRIM(IndexStr)<>''
  @ Row()+2,5 SAY 'Indexing file.'
  IF FILE("RAM:ITEMP.NDX")
    DELETE FILE RAM:ITEMP.NDX
  ENDIF
  INDEX ON &IndexStr TO RAM:ITEMP
ELSE
  IndexStr = ' '
ENDIF
RETURN


* Data filter program

PROCEDURE Set-Filter
Key = ' '
ERASE
@ 1,1 SAY 'MailList Filter'
@ 3,1 SAY "Current filter - " + FilterFld + " ";
          + FilterType + " " + FilterStr
FilterFld = FilterFld + DUPCHAR(RANK(' '),10)
FilterStr = FilterStr + DUPCHAR(RANK(' '),30)
FilterType = FilterType + DUPCHAR(RANK(' '),2)
@ 5,1 SAY 'Fields: LAST, FIRST, STREET, CITY, STATE,';
         +' ZIP, PHONE, and NOTES'
@ 6,4 SAY 'Enter name of field to filter on: ';
      GET FilterFld PICTURE '!!!!!!!!!!'
SET CONFIRM ON
READ
FilterFld = TRIM(FilterFld)
IF FilterFld <> ''
    @ 8,1 SAY 'Enter word(s) to search for: ';
          GET FilterStr PICTURE DUPCHAR(RANK('X'),30)
   READ
   FilterStr = TRIM(FilterStr)
   IF FilterStr <> ''
       @ 10,1 SAY 'Relational Operators:'
       @ 11,5 SAY '=  (equal)'
       @ 12,5 SAY '>  (greater than)'
       @ 13,5 SAY '<  (less than)'
       @ 14,5 SAY '<> (not equal)'
```

continued...

City Desk takes a Mega Bite

Out of Apple's Desktop Publishing Market



$149.95 (U.S.)

City Desk is a full featured Desktop Publishing Program designed with both the professional and amateur in mind. Now you have the power and flexibility to create high quality, professional looking documents. City Desk is for the serious users who demand the best, in products and results.

Version 1.1 Features

- Supports WordPerfect
- 140 Page manual created with City Desk
- Simple start Up exercise using text and graphics
- Automatic kerning and leading
- Powerful embedded command options
- Unlimited font changes in the text
- Flow text around graphics
- Any number of fonts on a line
- All preferences printers supported
- Prints IFF pictures
- Prints color pictures in gray scales
- Text and graphic editors included
- Headers or Footers
- Automatic page numbering
- Horizontal and vertical ruling with variable line weights
- Widow and orphan control
- Indents and outdents
- Left, right, center, or fill justification
- Reverse type and graphics
- Not copy protected!

Version 1.1 Now Shipping!

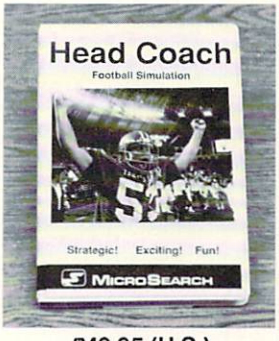
- Supports WordPerfect
- Supports PostScript
- Supports HP LaserJet+

Version 1.0 Owners Call for FREE Upgrade

(This page was created with City Desk and a PostScript Printer.)

Head Coach — Pro Football Simulation

If you've ever wished a computer football game could be more like a chess game... then its time you met the new Head Coach. Head Coach is a strategic game, not an arcade game. Playing Head Coach is as close to coaching the Pro's as you can get without signing a contract. You send in the plays, setting the strategy for those exciting long drives toward your opponents end zone. You call the plays the same way a coach calls plays. It's easy and fun to have the QB hand off to the Halfback and send him through the "three hole", by entering the simple command "RHB3". With Head Coach you can use the standard offensive or defensive playbooks or create your own. You can even design custom plays while the game is in progress!



$49.95 (U.S.)

- Have instant replays, even slow motion
- Show Stats while game is in progress
- Player injuries and substitute players
- Create realistic defensive alignments
- Returns fumbles and interceptions
- Call blocking assignments and snap count
- Display jersey number or player strength
- Computer can run either, neither, or both teams
- Create weather; wind, sun, rain, or snow
- Choose stadium type, name and surface

Now Shipping!

MICROSEARCH
9896 Southwest Freeway, Houston, TX 77074, USA, (713) 988-2818

Canadian Dealers: We now ship direct from Vancouver B.C. Call us for details. "At MicroSearch, we listen to our customers...carefully."
PostScript is a reg. TM of Adobe Systems Inc., WordPerfect is a reg. TM of WordPerfect Corp., HP LaserJet+ is a reg. TM of Hewlett-Packard Corp. Apple is a reg. TM of Apple Computer Corp.

```
        @ 15,5 SAY '<= (less than or equal)'
        @ 16,5 SAY '>= (greater than or equal)'
       @ 17,3 SAY 'Enter one: ' GET FilterType PICTURE 'XX'
      READ
       FilterType = TRIM(FilterType)
       IF FilterType <> ''
            Key = FilterFld + " " + FilterType + " " + "'";
                + FilterStr + "'"
           ? ' '
           ? 'Filtering file.'
           SET FILTER TO &Key
      ELSE
           SET FILTER TO
           FilterFld = ' '
           FilterStr = ' '
           FilterType = ' '
      ENDIF
   ELSE
        SET FILTER TO
        FilterFld = ' '
        FilterStr = ' '
        FilterType = ' '
   ENDIF
ELSE
    SET FILTER TO
    FilterFld = ' '
    FilterStr = ' '
    FilterType = ' '
ENDIF
Key = ' '
RETURN


* Data index program

PROCEDURE Set-Index
ERASE
@ 1,1 SAY 'MailList Index'
@ 3,1 SAY "Current index - "+IndexStr
IndexStr = IndexStr + DUPCHAR(RANK(' '),50)
@ 5,1 SAY 'Data field names:'
@ 6,5 SAY 'LAST, FIRST, STREET, CITY, STATE, ZIP,';
          + ' PHONE, and NOTES'
@ 7,1 SAY 'Enter index order: ';
      GET IndexStr PICTURE DUPCHAR(RANK('!'),50)
SET CONFIRM ON
READ
IndexStr = TRIM(IndexStr)
? ' '
USE &Dname&Fname
IF IndexStr <> ''
  ? ' Indexing file.'
  IF FILE("RAM:ITEMP.NDX")
    DELETE FILE RAM:ITEMP.NDX
  ENDIF
  INDEX ON &IndexStr TO RAM:ITEMP
ELSE
  ? ' Index removed.'
  IndexStr = ' '
ENDIF
Key = ' '
RETURN


* Add new records program - uses temporary file for data
* improved security.

PROCEDURE Add
? 'Opening data entry file.'
IF FILE('RAM:ADD.DBF')
  DELETE FILE RAM:ADD.DBF
ENDIF
COPY STRUCTURE TO RAM:ADD
SELECT SECONDARY
USE RAM:ADD
Key = ' '
DO WHILE (Key <> 'Q')
  APPEND BLANK
  Key = ' '
  DO WHILE (Key <> 'A') .AND. (Key <> 'Q')
    Key = 'A'
    ERASE
     @  4,29  SAY 'New Record Number:'
     @  4,Col()+1 SAY # PICTURE '#####'
     @  6,15  SAY 'First Name'
     @  6,Col()+1 GET First PICTURE '!!!!!!!!!!!!!!!'
     @  7,16  SAY 'Last Name'
     @  Row(),Col()+1 GET Last;
                PICTURE '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!'
     @  8,19  SAY 'Street ' GET Street
     @  9,21  SAY 'City ' GET City
     @ 10,20  SAY 'State ' GET State PICTURE '!!'
     @ 11,22  SAY 'Zip ' GET Zip PICTURE '#####'
     @ 12,20  SAY 'Phone ' GET Phone PICTURE '(###)###-####'
     @ 13,20  SAY 'Notes ' GET Notes
    SET CONFIRM OFF
    READ
     @ 18,30 SAY 'A = Add another name'
     @ 19,30 SAY 'Q = Quit adding'
     @ 20,30 SAY 'E = Edit this name'
     @ 21,39  GET Key PICTURE '!'
    SET CONFIRM OFF
    READ
  ENDDO
ENDDO
DISPLAY ALL
ERASE
Key = ' '
DO WHILE Key <> 'N'
  ? ' '
  ACCEPT 'Do you wish to print the new entries? (Y/N): ';
       TO Key
  Key = !(Key)
  IF Key = 'Y'
    DO Print
  ENDIF
ENDDO
ERASE
USE
SELECT PRIMARY
Key = ' '
ACCEPT 'If you wish to discard the new entries enter "Q": ';
       TO Key
Key = !(Key)
IF Key <> 'Q'
  ERASE
  ? 'Posting new entries to the main file.'
  APPEND FROM RAM:ADD
ENDIF
Key = ' '
RETURN


* Edit existing records

PROCEDURE Edit
Key = ' '
Macro = ' '
TargetFld = ' '
TargetStr = ' '
DO WHILE TargetFld <> 'Q'
  ERASE
  ? ' '
  ? ' '
  ? 'MailList Editor'
  ? ' '
  ? 'Fields: LAST, FIRST, STREET, CITY, STATE, ZIP,';
    + ' PHONE, and NOTES'
  ACCEPT 'Enter name of field to search (or Q to quit): ';
         TO TargetFld PICTURE '!!!!!!!!!!'
  IF TargetFld <> '' .AND. TargetFld <> 'Q'
    ? ' '
    ACCEPT 'Enter word(s) to search for: ' TO TargetStr
    ? ' '
    ? 'Searching'
    GOTO TOP
    LOCATE FOR TargetStr$&TargetFld
```

```
      Key = 'C'
      DO WHILE (Key <> 'Q') .AND. .NOT. EOF
       ERASE
        Key = 'C'
        SET CONFIRM ON
        @   4,28 SAY 'Record Number:'
        @ Row(),Col()+1 SAY # PICTURE '#####'
        @   6,17 SAY 'First Name ';
              GET First PICTURE '!!!!!!!!!!!!!!!'
        @   7,18 SAY 'Last Name ';
              GET Last;
              PICTURE '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!'
        @   8,21 SAY 'Street ' GET Street
        @   9,23 SAY 'City ' GET City
        @ 10,22 SAY 'State ' GET State PICTURE '!!'
        @ 11,24 SAY 'Zip ' GET Zip PICTURE '#####'
        @ 12,22 SAY 'Phone ' GET Phone PICTURE '(###)###-####'
        @ 13,22 SAY 'Notes ' GET Notes
        @ 19,28 SAY '^Q = quit this record'
       SET CONFIRM OFF
       READ
        @ 19,28 ERASE
        @ 19,28 SAY 'Q = quit searching'
        @ 20,28 SAY 'C = continue search'
        @ 21,37 GET Key PICTURE '!'
       SET CONFIRM OFF
       READ
        IF Key <> 'Q'
          @ 19,0  ERASE
          @ 19,35 SAY 'searching'
         CONTINUE
       ENDIF
     ENDDO
   ENDIF
ENDDO
Key = ' '
RETURN


* List file to video

PROCEDURE List
ERASE
? ' Rec #   Name - Last..................... First.....';
      + ' City.............. Zip..'
LIST ALL Last,First,City,Zip
ACCEPT 'Press <ENTER> to continue' TO KEY
Key = ' '
RETURN


* Print file master list.

PROCEDURE Print
ERASE
SET PRINT ON
? ' '
SET PRINT OFF
WAIT 'Press <RETURN> to continue'
ERASE
SET CONSOLE OFF
SET PRINT ON
* Printer set-up for Centronics 739
? CHR(27)+CHR(20)
* Printer set-up for Laser Jet
* ?? CHR(27)+'E'+CHR(27)+'&l0O'+CHR(27)+'(8U'+CHR(27);
    +'(s0p16.6h8.8v0s0b0T'
SET LEFT MARGIN TO 0
SET TOP MARGIN TO 0
SET BOTTOM MARGIN TO 0
GOTO TOP
? ' LAST                                          FIRST
STREET';
  +'                              CITY                    ST
ZIP';
  +'    PHONE             NOTES'
?
?
LIST OFF ALL TO PRINT
* Laser Jet reset command
* ?? CHR(27)+'E'
SET PRINT OFF
SET CONSOLE ON
? ' '
? 'Print complete.'
WAIT 'Press <RETURN>'
RETURN


* Label printer program for 3-up labels on 9" paper

PROCEDURE Labels
ERASE
Spacer = DUPCHAR(RANK(' '),61)
SET PRINT ON
? ' '
SET PRINT OFF
? ' '
? "Turn printer off, position three accross labels,";
  +" turn printer on."
ACCEPT 'Press <ENTER> to continue' TO Key
ERASE
SET CONSOLE OFF
SET PRINT ON
* Set 16 cpi font on Centronics printer
* then reverse feed the paper
? CHR(27)+CHR(20)
Counter = 12
DO WHILE Counter > 0
  ? CHR(27)+CHR(138)+CHR(27)+CHR(138)
  DEC Counter
```


continued...

```
ENDDO
Counter = 0
No:Printed = 0
SET TOP MARGIN TO 0
SET BOTTOM MARGIN TO 0
SET LINE COUNT TO 66
SET LEFT MARGIN TO 0
GOTO TOP
DO WHILE .NOT. EOF
   IF First = '                '
      Line1 = Last + $(Spacer,1,19)
    ELSE
      Counter = 18 - LEN(TRIM(First))
      Line1 = TRIM(First) +' '+ Last + $(Spacer,1,Counter)
  ENDIF
  Line2 = Street + $(Spacer,1,24)
  Counter = 38 - LEN(TRIM(City))
  Line3 = TRIM(City) + ", " + State + "   " + Zip;
        + $(Spacer,1,Counter)
  SKIP
  IF First = '                '
      Line1 = Line1 + Last + $(Spacer,1,19)
    ELSE
      Counter = 18 - LEN(TRIM(First))
      Line1 = Line1 + TRIM(First) + ' ' + Last;
            + $(Spacer,1,Counter)
  ENDIF
  Line2 = Line2 + Street + $(Spacer,1,24)
  Counter = 38 - LEN(TRIM(City))
  Line3 = Line3 + TRIM(City) + ", " + State + "   ";
        + Zip + $(Spacer,1,Counter)
  SKIP
  SET CONSOLE ON
  SET PRINT OFF
  No:Printed = No:Printed + 3
  @ 4,0 SAY 'Current label count:'
  @ ROW(),COL()+1 SAY No:Printed PICTURE '#####'
  SET CONSOLE OFF
  SET PRINT ON
  IF First = '                '
      Line1 = Line1 + TRIM(Last)
    ELSE
      Line1 = Line1 + TRIM(First) + ' ' + TRIM(Last)
  ENDIF
  Line2 = Line2 + TRIM(Street)
  Line3 = Line3 + TRIM(City) + ", " + State + "   " + Zip
  SKIP
  IF LEN(Line1) > 131
    Line1 = $(Line1,1,131)
  ENDIF
  ? Line1
  ? Line2
  ? Line3
  ? " "
  ? " "
  ? " "
ENDDO
SET PRINT OFF
SET CONSOLE ON
? ' '
? 'Print complete.'
ACCEPT 'Press <RETURN>' TO Key
Key = ' '
RETURN
```

•AC•

Amiga Pascal
Reviewed and Previewed

by Michael McNeil

When evaluating an implementation of Pascal language for the Amiga or any other computer, it is tempting to compare and contrast it with other programming languages. That is the wrong approach. There is no such thing as an all-purpose, all-powerful, all-capable computer language (or computer, for that matter).

Each language has its own strengths and weaknesses. Given the same programming task, equally expert programmers and the Amiga, the Macro Assembler version will probably run fastest, the C version will probably come in second, Amiga Pascal third and Amiga Basic last. This type of comparison is worthless. The real point is that some things can be done in Assembly more easily and quickly than in C, and vice versa.

Every programming language has a set of capabilities that make it particularly strong in some aspects, while sacrificing strength in other aspects. BASIC is commonly described as 'quick, but dirty'. C is often called a 'write-only' language — too difficult to comprehend when read. Assembly language has a reputation for speed, as well as difficulty.

Many things can be done quickly and easily with the Amiga Macro Assembler, Amiga (Lattice) C, Amiga Basic and Amiga (Cambridge) Lisp that can be done only with considerable difficulty in Pascal (on the Amiga or any other machine). Therefore, why hasn't Pascal become the Latin of programming languages?

Pascal is a computer programming language pioneered by Niklaus Wirth during the late 1960's and early 1970's. Wirth intended Pascal to be a good first language for people just learning to program. Because it was designed as a beginner's language, Pascal has a relatively small number of concepts to learn.

Wirth's concentration on easy implementation made the task of writing a compiler for Pascal relatively easy and, accordingly, Pascal quickly became available on many different machines. The real strength of Pascal is that it forces a programmer to write with a style that time has shown to be good, standard programming practice.

Amiga Pascal is actually MCC (Metacomco) Pascal 68000 and meets the ISO (International Standards Organization) standard 7185. 68000 represents the Motorola 68000 central processing unit used in the Amiga and other quality computers. In order to comply with ISO standard 7185 (and thus carry the name Pascal), Amiga Pascal had to meet or exceed a set of standards which define Pascal as a language.

One of the goals of standardizing a language is program portability. Ideally, you should be able to enter the source code of an error-free Amiga Pascal program on some other machine and compile it and run it without error. That is an admirable goal. Think about it. Can you do that with Amiga Basic?

If you try Amiga Basic, you can't run programs written in the earlier ABASIC without considerable changes. Both call themselves BASIC, both are languages for the Amiga, yet neither works with the other. Guess what? Within certain limits, an Amiga Pascal program can be entered, compiled and run on other machines without error and without changes.

Amiga Pascal, like any Pascal, forces you to do something that other programming languages don't force you to do. You must think the programming task through before you enter the code.

The design process of any programming project is critical to efficient code. Too many languages encourage you to design your programs well (without telling you what that means), while allowing you to develop bad habits. Not so with Pascal.

Here's an example. In Amiga Basic, you can begin your program with the "main" section of code. In that main section, you can make calls to subroutines. The code for those subroutines can then be entered, after the main section of code. Provided there are no syntax errors or logical errors, your program will run fine and produce valid results. It may even seem to run fast, but because you've been allowed to be inefficient by the language, you've built that inefficiency into your program. It may seem perfectly natural to place the subroutine code after the main section of your program, but you must remember how the computer sees the world.

Let's say you've written a program to compute a number raised to the power of three. Your main section gets the number to be cubed from the user and passes the

continued...

input to a subroutine that computes and returns the cube of the number. The main section then displays the result on the screen.

In execution, the Amiga follows your instructions precisely. It gets the number from the user and then encounters the instruction to pass that number to the subroutine. At this point, inefficiency creeps into your program.

The Amiga knows which instruction it is attempting to follow, so it can "remember" where to return to in your program after it finishes the subroutine. The Amiga doesn't know where to find the subroutine, though, so it must search for it.

Ever hear the phrase, "Begin at the beginning?" In most cases, that's exactly how a computer finds a subroutine. Beginning with the first line of the program, the computer examines each line of the program until it finds the subroutine. Sure, the examination process is very fast, but it does take some amount of time to examine each line. The sooner your subroutine is found, the faster your program runs. This example is only one of the many aspects of efficiency in programming.

Pascal gives you no option. You must enter your subroutines (called procedures or functions), followed by the main section of your program. If you try to switch the order, your program will not be accepted by the compiler.

Because you must enter your subroutines first, you must have a very good idea of exactly what they are, and where they fit in the overall scheme of your program. You have no option. You have to think the program through before you type it in.

Amiga Pascal measures up very well against the ISO Pascal standard. Perhaps too well. One of the unfortunate side effects of standardizing a language is that any version of the language inherits the shortcomings of the standard. As time goes on, the "standard" falls further and further behind the times. Considering the fact that the Pascal standard has been around since Wirth designed the language in the early 70's, it is not surprising that Amiga Pascal has no built in facility to explore the Amiga's powerful audio-visual features. You might expect Amiga Pascal to be standard Pascal, plus some added facilities to explore the unique features of the Amiga. Not so.

It's not that the developers didn't think about adding capability to Amiga Pascal. Version 1.0 includes the directive EXTERNAL (see table). The errata says the directive isn't implemented in version 1.0 and even if it were implemented, you would have to do some things that the manual doesn't mention (like buying the Macro Assembler just for Amiga.lib, which wasn't included with Amiga Pascal).

The next step is documentation. There is such a thing as too much documentation and Amiga Pascal made sure to avoid that shortcoming. The thin manual supplied with version 1.0 basically states that it is not intended to be a primer or tutorial of the Pascal language and that no prior knowledge of the Pascal language is assumed (I'm not sure if they really thought about that. If they assume you don't know Pascal, wouldn't tutorial-style documentation be appropriate?).

In any case, you should not try to learn Pascal from this manual. Many things relating MCC 68000 Pascal to the Amiga aren't stated very clearly, such as: You should enter your source code using the CLI Ed (or Edit if you prefer a single line editor) or some other editor. There is no "extension" required on the source code filename, but <filename>.pas is commonly used. The manual doesn't include instructions for using CLI or Ed, even if you figure out that you should use an editor.

Assuming you already know that Pascal is a compiled language, and that your source code must be compiled into object code, and that the resulting object code must then be linked to the Amiga operating system before the program can be run, you must know exactly how to go about accomplishing that task on the Amiga. You won't find a well-written description in the manual. Many unspoken assumptions are made. The errata helped, magazine articles helped some more and the Macro Assembler manual helped even more, but clarity was still lacking.

It should be easy to write instructions telling you how to run your program after it has been compiled and linked; the process is simple. While in a CLI window, type the filename of linked program file and press return. DO NOT type RUN before the filename. DO NOT type EXECUTE before the filename. Just type the filename. This tip might save you a day or two of trying to fix a perfectly good program.

Although the documentation is unexpectedly frustrating, you will eventually realize that what the manual states so incomprehensibly is also stated correctly. You just won't discover this fact until you have succeeded in compiling, linking and running a program on your own.

The manual proclaims itself a reference manual for MCC 68000 Pascal. An index would seem to be an appropriate item in a reference manual, whether it is assumed that the user knows Pascal or not. There is no index included here. Even if you know what you want to look up, it is going to take you quite some time to find it.

In all fairness, I am writing this article from the vantage point of hindsight. I suspect that the documentation problem lies somewhere in the original Amiga development effort. It is difficult for someone to develop documentation for one part of a large project, while someone else develops the documentation for another part, and yet a third party develops the documentation that draws it all together. Some ambiguity in each part should be expected.

continued...

UNCLE D'S CON*SOUND*TRATION

FOR KIDS 4 & UP

A game for children using the *AMIGA's* great sound and graphics. In this modern version of the old favorite concentration game, your child must match a picture with the correct sound... sounds such as musical instruments, home sounds, animal noises, and environmental sounds.

One or two players.
Four play levels.
Three different games... PICs, ABCs, 123s.
Randomly generated gameboards.
Easy to recognize digitized sounds and colorful artwork.
Helps build ear–eye–hand coordination.
Requires no reading skills.

WARNING!
May be too hard for most adults!
A challenge for young & old alike.

UNCLE D'S CON*SOUND*TRATION	$39.95
ALOHAFONTS VOL 1	$19.95
ALOHAFONTS VOL 2	$19.95

At your dealer now.

OR ORDER DIRECTLY FROM:

ALOHAFONTS™

AlohaFonts, P.O. Box 2661, Fair Oaks, CA 95628–2661

Please include $2.50 shipping and handling for each item.
California residents please add 6% sales tax.

Good news, bad news

Commodore Amiga is working on an upgrade to Amiga Pascal. They were kind enough to let me work with a copy of what they have so far. The EXTERNAL directive works, but not as I expected from the documentation. No new documentation was provided with the upgrade they sent to me. I was still saddled with the old stuff, just fewer errata to be concerned with. So far, the upgrade only goes as far as trying to make Amiga Pascal perform in accordance with the documentation for the original version.

The direction that upgrade plans will finally take has not been decided. Those plans will be announced soon. I cast my vote for a full-blown Amiga Pascal, complete with the tools to enable the "good stuff," and good, thorough, tutorial-style documentation. Commodore also hinted that a reunion between Commodore and Metacomco might be part of the upgrade announcement.

At Commodore's suggestion, I tried calling Metacomco to ask them about the status of their other Pascal package for the Amiga which is independent of Commodore's Amiga Pascal. Apparently, Metacomco has gone home to England; they no longer have an office in the U.S.. If you want to call Metacomco, call the operator at 800–654–9812 and let her know that you are trying to dial 408–438–7201. You'll be given Metacomco's overseas number and address.

In conclusion, Amiga Pascal is a very good implementation of the Pascal language. The real weaknesses of Amiga Pascal version 1.0 are its failure to explore the special power of the Amiga and poor documentation.

(Note: If you are taking Pascal as a course in school, the following table may help convince your instructor that Amiga Pascal is suitable for homework assignments. Instead of competing for time on the school's system, you could be doing the assignment at home on your Amiga.)

•AC•

Table of Amiga Pascal

RESERVED WORDS, SYMBOLS, IDENTIFIERS

AND	ARRAY	BEGIN	CASE	CONST
DIV	DO	DOWNTO	ELSE	END
FILE	FOR	FUNCTION	GOTO	IF
IN	LABEL	MOD	NIL	NOT
OF	OR	PACKED	PROCEDURE	
PROGRAM	RECORD	REPEAT	SET	THEN
TO	TYPE	UNTIL	VAR	WHILE
WITH	+	.	=	>
{	-	,	<>	(
}	*	;	<	)
^	/	:	<=	[
..	:=	'	>=	]
FORWARD	ABS	ARCTAN	BOOLEAN	CHAR
CHR	COS	DISPOSE	EOF	EOLN
EXP	FALSE	GET	INPUT	INTEGER
LN	MAXINT	NEW	ODD	ORD
OUTPUT	PACK	PAGE	PRED	PUT
READ	READLN	REAL	RESET	REWRITE
ROUND	SIN	SQR	SQRT	SUCC
TEXT	TRUE	TRUNC	UNPACK	WRITE
WRITELN				

Notes

- Compile-time LIST option, errors/warnings interspersed with source code in resulting file.
- Implementation: level 0 of ISO 7185 (BS 6192) standard
- MAXINT = 2147483647, -MAXINT = -2147483647
- Data Types supported:
  - Simple
    - Integer
    - Real
    - Character
    - Boolean
  - Structured
    - Enumerated and ordinal sub-range(s) of
    - Sets
    - Arrays
    - Records
    - Pointers
    - Files
- Sets can contain up to 250,000 elements.
  - Set operators:
    - = test on equality
    - <> test on inequality
    - <= test left set is subset of right set
    - >= test right set is subset of left set
    - IN test for membership in set
    - * form intersection of two sets
    - + form union of two sets
    - - form difference of two sets
- Extensions to ISO standard:
- RESET and REWRITE extension allowing internal files to access external files. External file name(s) may be passed at run time.
- INCLUDE, allowing source code of modules to exist in their own files, for inclusion at compile time.
- EXTERNAL, allowing programmer to reference (the object code of) other programs as modules of this program. Not implemented in Amiga Pascal version 1.0.

AC-BASIC Compiler

Overview—Part II

by Bryan Catley

Ed Note: The first article in this series, "AC-BASIC Compiler, Part I" (V 2.9, p. 64), listed the current Absoft AC-BASIC compiler as release 2.1. The current release from Absoft is actually version 1.2. We apologize for any inconvenience caused by our error.

Last time, we discussed the AC/Basic package. We also discussed its compile time options, extensions to the AmigaBasic language, metacommands and the (very few) differences from the AmigaBasic interpreter. Finally, we touched upon some of the more annoying current problems and finished up by recommending that serious Basic programmers (who don't mind a few frustrations) purchase the compiler, while others wait for a release or two. (We also stated that ultimately, every AmigaBasic programmer will want a copy of this compiler in his library).

Compiler Output

Even though the compiler is very rapid, it manages to keep up a continuous stream of output to keep you advised of its progress. This output is directed to the monitor and is referred to as compiler statistics. Of course, the primary output is the machine language version of the program which is written to the current directory.

The statistics start immediately with "0: file-name," where "filename" is the name of the program you are compiling. This initial line is followed by: "1: nnnn," where "nnnn" is a running count of the program lines processed. (N.B. this figure bears no relationship to any Basic line numbers present in the program).

Once a single pass has completed, the following statements will be displayed on the screen:

```
0: file-name
1: Symbol table complete
     Memory usage:
       Labels  nnnnnnnn bytes
       Symbols nnnnnnnn bytes
       Total   nnnnnnnn bytes
       Excess  nnnnnnnn bytes
       Source  nnnnnnnn lines
2: 00000
```

"Total" and "Excess" provide the most useful information. These figures allow you to adjust the amount of work space (set by the slider on the Control Panel—remember?) made available to the compiler. If you make use of the multitasking ability of the Amiga, these values will help you utilize your memory to its best advantage. Should you single-task your Amiga, these statistics may be considered general information.

The second pass now starts and, once again, a running count of the program lines appear against the 2:. Assuming there are no errors, the compiler issues the message: "2: Object file complete."

Next the third, and final, pass starts. This pass is actually more of a linker pass than a compiler pass. In fact, messages appear to indicate the linking of MAIN and each subprogram (by name) you have in the program.

When this linking pass has been completed, the following additional messages are displayed:

```
3: Program file complete: nnnnnn bytes
     Stack Size: nnnnn bytes
     Elapsed time: m:ss = lines/minute
```

Note the program size is only the amount of memory required to store the program; it does not include memory required by dynamic arrays, custom screens and windows, file buffers, etc.

The stack size is of particular importance to CLI users. The default stack size for any program is 4000, and if a program requires more than this specified amount, the result is almost always a system crash. Therefore, CLI users must use the STACK command appropriately. Workbench users need not concern themselves with stack size, since it is all handled automatically.

Once the compile is complete, an "iconed" program results, which can be run by double-clicking on the icon or by typing its name at a CLI prompt. Neither the AC/Basic compiler nor the AmigaBasic interpreter need be present! However, if you did NOT select the R compile time option, you will need the run time library (if you did select the R compile time option, the program is COMPLETELY independent).

If errors are detected during the compile, 2: is bypassed, and 3: becomes an error report. A compile time error report which looks like:

```
***** error in line nnnnn (xxxx): qq - text
```

nnnnn	is the physical line number
xxxx	is a reproduction of a portion of the erroneous program line
qq	is an error number which may be looked up for further information in the provided documentation
text	is a brief description of the error

continued...

My biggest complaint with all this info is that "xxxx" is usually too brief to be recognizable. It becomes very difficult to find and correct the statement in error, especially when the error occurs towards the end of a large program! Further, most original AmigaBasic programs don't use line numbers, so "nnnnn" may not be very useful, unless you already have a complete listing and are willing to do some mathematics.

Very frequently, it is necessary to recompile with a full list directed to disk, which is then "browsed" or printed. (A situation which is not that uncommon in the main frame world).

I think it may help to expand "xxxx" to show the entire program line, even if it means a two line error message. Other than that slight help, I'm not sure what else might be done to improve this particular situation.

The disk files produced by the compiler are named as the program being compiled, but the executable program is given a ".run" suffix and any list file is given a ".lst" suffix.

This ".lst" file may be printed by selecting "Print" from the compiler's project menu. N.B. At this time, the ".lst" files are not given icons, making them very easy to forget!

Using the Compiler for the First Time

When you first use the AC/Basic compiler, you might as well compile the sample program "Hello" right from your initial backup disk, as the documentation suggests. This program uses graphics to "rotate" the letters in the word "ABSOFT," and may be used to try out all the various compiler options. Additionally, running the compiled version, and then running the same program under the interpreter will give you a pretty good idea of how much faster compiled programs actually run!

Installing the Compiler for Day to Day Use

Almost certainly, you will want to set up a special AC/Basic work disk, whether it be a special Workbench disk or (as I have things set up) an individual work disk (it's up to you and your hardware configuration). This disk will be your "working" disk. Appendix H of the AC/Basic documentation provides all sorts of information for custom installations, including hard disks. The only thing I would add to this comprehensive appendix is a suggestion that you, by all means, should include a copy of AmigaBasic on ANY disk on which you have AC/Basic installed.

When you start compiling your own programs, you may very well end up with perfectly valid syntax errors that you never expected to see! Remember, the interpreter only looks at statements as they are executed; if a statement is never executed, the interpreter can never check it. However, the compiler checks EVERY statement within the program, giving syntax errors wherever appropriate!

Compiler Times

As indicated a little earlier, AC/Basic is really a combined compiler and linker. Passes 1 and 2 do the compiling, while pass 3 does the linking. All this work is done very rapidly. Don't expect to be going for a cup of coffee while waiting for a compile to complete with AC/Basic!

The following tables represent various compile times for the given 332 line sample program ("Hello") and the 909 line "HouseInv" program (see Amazing Computing Volume 2 Number 4). Since the compiled programs may be executed by double-clicking, information produced by the INFO Workbench option is also included.

	Opt	CSize	CStack	Size	Stack	Time	Lns/Min
Program Hello (332 lines)	CT	6912	4468	7648	4468	0:28	711
	CNT	8596	4468	9332	4468	0:29	686
	CNTR	8596	4468	50852	4468	1:29	223
	T	6096	4468	6832	4468	0:25	796
	TU	6072	4876	6808	4876	0:28	711
	RTU	6072	4876	48328	4876	1:23	240
	NT	7780	4468	8516	4468	0:32	622
	NRT	7780	4468	50036	4468	1:27	228
Program HouseInv (909 lines)	CNT	53264	4706	54000	4706	1:06	826
	CNTR	53264	4706	95520	4706	2:14	407
	NT	48304	4706	49040	4706	1:06	826
	NRT	48304	4706	90560	4706	2:16	401
Program HouseInv (Using DIM STATIC where possible)	CNT	53664	6730	54400	6703	1:05	839
	CNTR	53664	6730	95920	6703	2:16	401
	NT	47920	6730	48656	6703	1:07	814
	NRT	47920	6730	90176	6703	2:15	404

• Since "HouseInv" makes use of ON MOUSE and ON MENU statements, the use of the N compile time option is mandatory.

• The CSize figures obviously do not include the size of the run time library when the R compile time option is chosen.

• "HouseInv" contains a number of static arrays, and one array which must be dynamically allocated, making use of the U compile time option impossible.

(CSize and CStack represent statistics produced by the compiler; ISize and IStack represent information produced by the Workbench INFO option).

Run Time Errors

As with any language, the fact that a program compiles correctly is no guarantee that it will run correctly. AmigaBasic is no exception. To handle this likely situation, AC/Basic provides a number of run time error messages, rather than just terminating the program in a rather ungracious manner. The format of these messages is as follows:

```
***** Error n <at lllll>
Click Mouse
```

"n" is an error number (check Appendix F in the supplied documentation for more information) and "lllll" is a physical line number within the program. This last format is only used when the "N" compile time option has been chosen.

Since it can be a somewhat difficult to find a specific physical line number within a program, it is probably advisable to develop and test a program using the interpreter. Then,switch to the compiler only when you are sure the program is functioning the way it should. (But, be prepared to encounter some errors which the interpreter didn't catch!)

AC/BASIC Compiler Bugs

As I have already indicated, I have encountered a number of bugs in the process of putting this review together. It is only fair that I share them with you:

Compile Time Bugs

• If the compiler runs out of work space during the compile, it leaves its work files open. This quirk means these work files cannot be deleted, nor can the compiler be used again. A re-boot is the only way to get rid of the offending files.

• I received a syntax error on a statement using several pairs of parentheses when, in fact, there was no error.

• If a program is compiled without the U option, or without using DIM STATIC, and is then recompiled with either of these options, the INFO stack size information is not updated as it should be. This difficulty means the program will probably crash if started from the Workbench. To correct this situation, click in the INFO stack size box, update it and save. Your only other option is to erase the program from disk before recompiling (This bug may be an Amiga bug of some sort, rather than an AC/Basic bug . . . in any event, you need to be aware of it).

continued...

Run Time Bugs

This category is, by far, the most interesting!

• Multiple mouse clicks (in a gadget, for example) are sometimes necessary before the click is recognized.

• A program making extensive use of the mouse may freeze at some point during its use. Sometimes, my mouse would freeze quickly, sometimes it would take awhile. Sooner or later, though, it would freeze, forcing a re-boot. This problem only occurred with one program.

• If a program defines a menu with a certain number of items, and later re-defines that same menu with fewer items, the excess menu items from the original menu remain. Selecting one of these excess items crashes the system.

• A number of demonstration programs, which made extensive use of operating system calls, resulted in a blank window and screen (i.e. no icons) when terminated. There was no option, but to re-boot. A variation of this problem was a "Software Error" requester. The net result, however, was the same.

• When using animation, collision detection occurred at the edge of the window, rather than where it was supposed to happen.

• Mouse clicks were not detected correctly when multiple windows are open.

• The Text& function does not recognize screen positioning from a previous PRINT PTAB statement. For example, with the following two statements:

```
PRINT PTAB(20);:CALL
Text&(RP&,"Abc",3)
```

"Abc" ends up in column one of the following line, rather than 20 pixels from the left of the current line.

• Not a bug, but... The SORTSUBS utility program (which rearranges subprograms scattered throughout a program), and which is provided as part of the package, spends much time manipulating statements without any screen output being generated. With a larger program, it is very easy to think the whole machine has just died! I believe an occasional "Working..." message directed to the screen would be very reassuring for many users.

Absoft is aware of just about all of the above problems, and many of them, if not all, will be corrected in an upcoming maintenance release (which, I believe, will be free to REGISTERED users).

While it cannot be classified as a bug, there is one other quirk which fully concerns me — the size of the compiled module. As far as I am concerned, you should save some memory when you compiling a program which has previously been used with an interpreter. After all, there is no need for the memory used by the interpreter itself, and surely the compiled code should require less memory than the combined source code and the interpreter? With the case of AC/Basic compiled programs, this case apparently does not hold true. Indeed, some programs which just fit (and work) with the interpreter, will run out of memory when a compiled version is executed.

I would gladly accept increased compile times, if the size of the resulting program module could be reduced.

AC/BASIC Documentation

While not perfect, the documentation which accompanies AC/Basic is excellent. Chapters 1 to 6 document the compiler, how to use it, extensions, etc. Chapters 7 to 18 are an AmigaBasic reference manual, sequenced by usage. Appendices A through I cover a number of pertinent topics, including compile and run time error messages, installation notes, calling assembler language subroutines, etc. Remember, many complete examples are sprinkled throughout the manual and all of them are on the distribution disk in drawers labeled by Chapter number. My biggest complaints regarding the documentation are really nit-picking comments, but they are worth noting.

First, the pages are very busy. Lots of information is crammed onto each page and it is often difficult to find the exact information you are looking for. Second, the index must have been created from an earlier version of the documentation. Many times, I found information on a different page from that indicated in the index. Third, and last, dividers between the compiler section, the reference section and the appendices would really be beneficial and appreciated.

Overall, you will be hard pressed to find better, more complete documentation for any product.

Some Final Comments

The AC/Basic compiler has the potential to be one of the most important products ever released for the Amiga. It is not quite there yet, but once the outstanding bug list has been reduced to an absolute minimum (remember, all compilers on all computers have outstanding bug lists for most of their lives), and once the size of the compiled code has been reduced, every serious AmigaBasic programmer will want to own a copy of AC/Basic.

When you buy AC/Basic, REGISTER your purchase immediately. Call Absoft's Technical Support number when you have problems, but please don't expect Absoft to debug your program for you!

AmigaBasic programmers could be in a very enviable position. Very few others will be able to combine the advantages of interpretive development with the independence of compiled production code. Use it to your advantage!

•AC•

AmigaBASIC Structure

How lines of BASIC code are represented internally by AmigaBASIC and more...

by Steve Michel

This article takes another step down the AmigaBASIC road by shedding light on structure. This discussion is not definitive, but rather a smattering of details from the innards of the Amiga.

The AmigaBASIC DECODER, shown in Listing 1, illustrates how lines of BASIC code are represented internally by AmigaBASIC. DECODER reads and analyzes a specified program file from disk. Earlier versions of DECODER tried to analyze the program from RAM memory while the program was resident in the AMIGA. This method proved too slow and cumbersome, though, because the architecture of the Amiga is so radically different from most other machines.

Early PET computers loaded BASIC programs into memory starting at a constant address. This constancy made life a lot easier for both the operating system and the explorer. The Amiga allocates memory dynamically to different tasks as it sees fit. AmigaBASIC is no exception. One consequence of this process is that the starting memory location of a loaded BASIC program may be anywhere in memory, depending on such factors as number of disk drives, whether a CLI window is currently available or how many other multi-tasked programs are active.

When DECODER runs, it asks for the name of the file to be examined, whether to output the whole file or just the variable table and, finally, if screen or printer output is desired. DECODER then displays program information in a format similar to Figure 1.

The example output consists of twelve lines of numbers that show the internal representation of each of the twelve lines of the AmigaBASIC program listed in Figure 1. DECODER produces only the lines of numbers. The reference line numbers at the beginning of each line and the explanatory line description notes below the number lines were manually added after DECODER finished running. The output in Figure 1 reveals that the structure of each line of AmigaBASIC adheres to the following general format.

• Each line starts with a two byte descriptor. The first byte names the length of the line in bytes, including descriptor bytes and end of line markers. The second byte specifies the number of spaces to be INDENTED when the line is listed to the LIST WINDOW or printer. This arrangement represents good memory management — indented lines do not consume bytes of memory for each space, but rather a single byte counts all indentation. For example, if a line is indented ten spaces and actual spaces are used, nine bytes of memory are wasted. In a large program, such squandering of memory can be costly.

• The tokenized form of the line occurs next. If you are not familiar with tokens, a token is a number that represents a BASIC keyword. For example, the keyword PRINT is not stored in memory as the letters 'P' 'R' 'I' 'N' 'T' (which would consume 5 bytes of memory), but rather is converted to a specific number (let's say 158 for the sake of illustration). This keyword-to-number conversion process occurs as each line is entered into the LIST WINDOW. This number, through another translation process, produces the effects of the PRINT statement when the program is executed. Such tokenization accomplishes two important points: 1) memory conservation, by using one or two bytes of memory for the token, instead of one byte for each letter. 2) speeding up the execution process.

In previous versions of Commodore Microsoft BASIC, the tokens were one byte in length, similar to lines five (FOR = 148) and six (IF = 152). AmigaBASIC, however, has over half of its keywords represented by two byte tokens, similar to line three (CLS = 248 131). I'll take the easy way out and say that the reason for two byte tokens is beyond the scope of this article (In other words, I don't have the foggiest idea why two byte tokens are used. A single byte would easily handle the 196 possible keywords. Maybe someone else out there would be kind enough to fill us in on this one. Is anyone at Microsoft listening ?).

• Finally, each line ends with two consecutive zero bytes shown as 'EOL.' Incidentally, the term 'SP' indicates a space and 'PTR' indicates a pointer.

A closer look at Figure 1 reveals a few other facets of AmigaBASIC:

continued...

• BLANK LINES - as shown in lines one and twelve; use four bytes of memory (true for all blank lines).

• VARIABLE AND LABEL NAMES - are not stored directly in the BASIC text area, but rather are stored in a variable table at the end of the BASIC text. The end is denoted by two consecutive zeros in the line LENGTH and INDENTATION locations.

In the AmigaBASIC program, variable and label names are represented by either three or four byte pointers. These pointers determine which variable or label will be accessed in the variable table. Three byte pointers are used for variables and labels, while four byte pointers are used for labels following THEN or GOTO statements. For example, the variable X in lines four and five is represented by the pointer 1 0 1. The label BEGIN in line two has the pointer 2 0 0, while ALLDONE in line ten has the pointer 2 0 4.

The four byte pointer in line six (3 0 0 4) indicates that the label ALLDONE follows a THEN statement. The first byte of the pointer indicates the pointer's TYPE. 1 denotes a VARIABLE pointer, 2 is a LABEL pointer and 3 is reserved for LABELS following GOTO or THEN statements. The last two bytes of the pointer indicate the variable's POSITION in the variable table. AmigaBASIC uses this POSITION value to determine which variable or label is currently in use.

A pointer of 1 0 1 is interpreted as a VARIABLE (first byte = 1) occurring in the second location of the variable table (0 1). Remember, the first location denotes position zero (0 0). The use of two bytes for the table position means that an AmigaBASIC program has a limit of 65536 (256 * 256) total variables and labels (I think we can all probably live within that restriction).

• The VARIABLE TABLE is an integral part of the program which is loaded and saved as part of the BASIC program. When a label or variable name is entered into a program, its name is added to the current end of the table. Each variable in the table is preceded by a single byte (denoting the length of the variable name) and followed by the ASCII codes of the characters used in the variable name. Any variable type descriptors ($, %, #, !, etc.) are also stored at the end of the variable name. In the sample program in Figure 1, the variable table (BEGIN, X, Y, Z, ALLDONE) actually look like: 5, 98, 101, 103, 105, 110, 1, 120, 1, 121, 1, 122, 7, 97, 108, 108, 100, 111, 110, 101. These values are in decimal form and can be found on page A-2 of the AmigaBASIC manual.

This table seems rather straight-forward, but some strange things may happen. Variable names are placed in the table as soon as they are entered into the program via the LIST WINDOW editor. Once a variable name has been placed in the table, it COULD be there forever. Even if a variable is completely removed from the program, it is still retained in the variable table. Theoretically, you could write a program with 100 variables, each using the full 40 character name length, thereby creating a variable table of 4100 bytes. All but a single REM statement could then be deleted from the program, but the variable table would still be 4100 bytes long !!! Variable and label names are probably retained for one very important reason... speed. More about this point in a moment.

Fortunately, there is a way to "clean-up" the variable table. The trick is to save the BASIC program using the ASCII option, then re-load the newly saved ASCII version and re-save the program as a regular BASIC program. This process removes any variables that are no longer being used and puts the variables in the order they are encountered in the program, from top to bottom (chronologically, if you will).

Saving a program in ASCII format produces an "un-tokenized" version of the program that can be loaded into a word processor and edited like any other document. In this case, the variable table is not saved with the file. When this ASCII file is loaded, AmigaBASIC must rebuild the variable table

from scratch, resulting in a nice, "clean" variable table. To save an AmigaBASIC program as an ASCII file, save the file as follows:

SAVE "filename", A

Why did Microsoft opt for allowing the variable table to get cluttered with discarded variables? Removing a variable from the table requires moving all the variables up one position, filling the deleted variable's former location. This task requires that every pointer within the BASIC text be updated to reflect its new position within the recently revised variable table. This job is a big one and could take a considerable amount of time, depending upon the size of the BASIC program. In this trade-off between speed and a "clean" variable table, Microsoft opted for speed. Since we are now aware of this situation, saving as an ASCII file and re-loading every now and then to help un-clutter is a good habit to get into when developing a new program.

Now that we've seen the inner workings of AmigaBASIC, what can we do with our new found knowledge? One useful application would be a compacting program to run through a program, stripping REMs, apostrophes and blank lines, while left-justifying each line along the way. Sure, this solution runs contrary to good, structured programming style and documented code, but faster execution speed or a reduction in size may well warrant such action. A compacting program is provided in Listing 2.

This program prompts the name of the file to be compacted and the name of the new compacted file to be constructed. A new file is created, so the original is left intact for further work. COMPACTOR itself is heavily documented, but a few comments and explanations may be helpful.

Each program file contains two leading bytes, 245 ($F5) and 0 ($00). These bytes identify the file as an unprotected AmigaBASIC program and are copied directly over to the new file (These bytes are changed to 244 ($F4) and 194 ($C2) when the BASIC program is saved as a protected file. . . but that is another story).

continued...

Figure 1

Original AmigaBASIC program:

```
1)
2)   begin:
3)    CLS
4)    x = 1: y = 5
5)    FOR z = x to y
6)     IF SIN(z) = 0 then alldone
7)     print SIN(z)
8)    NEXT z
9)
10)  alldone:
11)   END
12)
```

Decoded program:

LENGTH INDENT EOL = end of line SP = space PTR = pointer

	V	V	
1)	4	0	0 0 (blank line) EOL
2)	8	0	2 0 0 58 0 0 begin PTR : EOL
3)	6	2	248 131 0 0 CLS EOL
4)	20	2	1 0 1 32 234 32 18 58 32 1 0 2 234 32 22 x PTR SP = SP 1 : SP y PTR = SP 5 0 0 EOL
5)	22	2	148 32 1 0 3 32 234 32 1 0 1 32 229 32 FOR SP z PTR SP = SP x PTR SP TO SP 1 0 2 32 0 0 y PTR SP EOL
6)	23	4	152 32 181 40 1 0 3 41 32 234 32 17 32 230 IF SP SIN (z PTR) SP = SP 0 SP THEN 32 3 0 0 4 0 0 SP alldone PTR EOL
7)	12	4	172 32 181 40 1 0 3 41 0 0 PRINT SP SIN (z PTR) EOL
8)	9	2	169 32 1 0 3 0 0 NEXT SP z PTR EOL
9)	4	0	0 0 (blank line) EOL
10)	8	0	2 0 4 58 0 0 alldone PTR : EOL
11)	6	2	248 143 0 0 END EOL
12)	4	0	0 0 (blank line) EOL

VARIABLE NAMES TABLE

begin	x	y	z
alldone			

COMPACTOR opens two program files, reading from the file to be compacted and writing to the compacted file. Beginning at the label LOOP: (if it is not the end of the BASIC text), a complete line from the first file is read into the array byte$(). This line is then scanned for BLANK lines, leading apostrophes and embedded REMs or apostrophes. Once a line has been analyzed, the byte$() array is adjusted at label SETUP.LINE to contain the correct information for the newly formatted line. This line is then written to the second file.

This process continues until the end of the first BASIC file is reached. At that point, the label END.OF.BASIC takes over and correctly finishes up the second file's end of basic markers. FINISH.UP copies the variable table and the icon for the BASIC program, so that a "clickable" compacted program is produced.

The number 175 is represented differently in two different places (as 174+1 and once as 7*25). This discrepancy is necessary because the number 175 also represents the token for the keyword REM. This duplicity confuses COMPACTOR because the number 175 also appears internally as a 175. Thus, COMPACTOR does not know the difference between a 175 that stands for a REM token and a 175 that stands for the number 175. The workaround assures that any programs that include the constant 175 are "fixed" in some way. Either method, 174+1 or 7*25, works fine. Both are illustrated to show that several options are available to alleviate this problem.

The last oddity MICROSOFT included (just to drive us nuts!) deals with the byte counters that appear in COMPACTOR. The early versions of COMPACTOR worked only about 50 percent of the time and the problem wasn't easily apparent. Everything looked fine. A flip of a coin seemed to determine whether the compaction was successful or not. The solution was finally tracked to the area between the end of BASIC text (indicated by two consecutive zeros) and the variable table discussed above. For whatever reason, programs containing an odd number of bytes of BASIC text start the variable table immediately after the end of BASIC. Programs with an even number of bytes of BASIC text insert a single byte between the end of BASIC and the beginning of the variable table.

This strange fickleness meant that the early version worked if an odd to odd or even to even byte compaction occurred because the compacted file would follow the same structure as the uncompacted file. If an odd to even or even to odd byte compaction occurred, some mighty strange things happened. Hence, the fifty-fifty success ratio.

This version of COMPACTOR counts the number of bytes coming in from the original file. If an even number is counted, the extra byte before the variable table is read and thrown away. If an even number of bytes has been written to the compacted file, an extra byte is added before the variable table (I used the ASCII value of the letter "J" for a friend who first suggested the even-odd possibility). The value of the spacing byte is irrelevant; it just has to be there! This phenomenon may be involved with something called 'even word alignment,' needed for certain modes of the 68000 microprocessor. When COMPACTOR runs, the original file length of 4764 bytes is compacted to only 2920 bytes (approximately 40% smaller).

This article has just scratched the surface of AmigaBASIC, but I hope it has provided a start for further explorations. Good luck.

Listing 1

```
REM AMIGABASIC DECODER
REM By Steve Michel

decoder:
  dc$ = "\                    \ "
  ON ERROR GOTO error.msg
  CLS: PRINT: INPUT "Enter filename to decode"; filename$
  PRINT: PRINT "Looking for file..."
  OPEN "I", #1, filename$
  byte$ = INPUT$(1,#1): byte$ = INPUT$(1,#1): bytecntr = 2
  PRINT: PRINT "Whole listing of just Variable table (W/V)"
getwv:
  wv$ = INKEY$: IF wv$ = "" THEN getwv
  wv$ = UCASE$(wv$): IF wv$ <> "W" AND wv$ <> "V" THEN getwv
  PRINT: PRINT "Screen or Printer (S/P)"
getsp:
  g$ = INKEY$: IF g$ = "" THEN getsp
  g$ = UCASE$(g$): IF g$ <> "S" AND g$ <> "P" THEN getsp
start:
CLS: max = 19: WIDTH 80
IF wv$ = "V" THEN
  PRINT: PRINT " BE PATIENT.  READING PROGRAM."
  LOCATE 6,2: PRINT "READING LINE # ";: linecntr = 0
END IF
loop:
  byte$ = INPUT$(1,#1): bytecntr = bytecntr + 1
  linelength = ASC(byte$)
  IF linelength = 0 THEN listvartable
  value = linelength
  IF wv$ = "W" THEN GOSUB printvalue
  FOR j = 2 TO linelength
  byte$ = INPUT$(1,#1): value = ASC(byte$)
  counter = counter + 1: bytecntr = bytecntr + 1
  IF wv$ = "W" THEN
    IF counter = max THEN GOSUB indent
    GOSUB printvalue
  END IF
  NEXT j
  IF wv$ = "V" THEN
    linecntr = linecntr  + 1
    LOCATE 6,17: PRINT USING "####"; linecntr
  ELSE
    GOSUB blankline
  END IF
  GOSUB pause
  GOTO loop

printvalue:
  IF g$ = "P" THEN
    LPRINT USING "### ";value;
  ELSE
    PRINT USING "### ";value;
  END IF
  RETURN

indent:
  max = 17: counter = 0
  IF  g$ ="P" THEN
    LPRINT: LPRINT
```

continued...

Avatex 1200

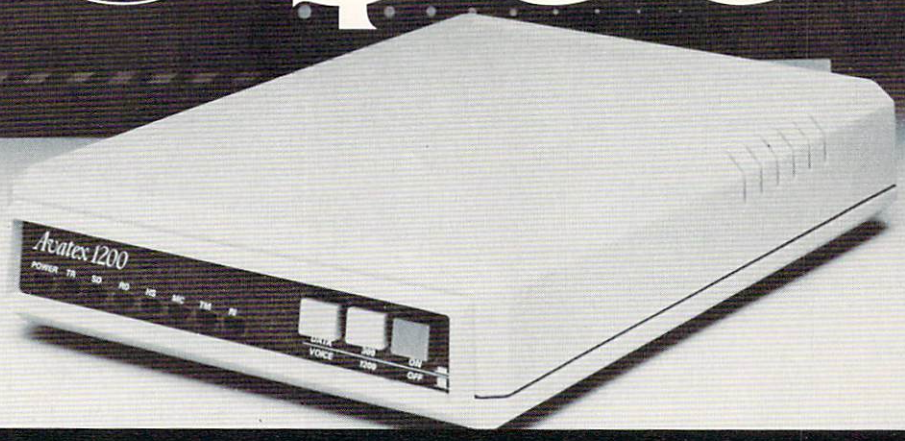
MODEM

THIS MODEM WORKS WITH ANY AMIGA.

- Direct Connect
- External AC Adapter
- 8 LED Status Indicators
- Auto Answer and Dial
- Hayes Compatible AT Command Set
- Bell 103/212A Compatible
- Tone or Pulse Dialing
- Telephone Connection Jack
- Standard DB25RS 232-C Connector
- FCC Approved

COMPUTER SHOPPER Best Buy OF THE YEAR

$85



FREE! Communication Software & Compuserve Access Time with each MODEM.

We give you a great price . . .

+ TOLL FREE ASSISTANCE
+ IMMEDIATE DELIVERY
+ NO CREDIT CARD SURCHARGE
+ NO SERVICE OR HANDLING FEES

We lowered our price.

Thousands of people are now enjoying this high quality Modem. It is full of features and works with any computer system. We are offering this special price for a limited time only.

FREE SHIPPING!

MEGATRONICS

To Order

CALL FOR FREE CATALOGUE
CREDIT CARDS VERIFIED FOR YOUR PROTECTION

800-232-6342

INSIDE UTAH (801) 752-2642 FAX (801) 752-8752

We'll beat any advertised price.

MEGATRONICS, INC., P.O. BOX 3660, LOGAN, UTAH 84321

```
    LPRINT "          ";
  ELSE
    PRINT
    PRINT "          ";
  END IF
  RETURN

blankline:
  IF g$ = "P"  THEN
    LPRINT: LPRINT: LPRINT
  ELSE
    PRINT:PRINT: PRINT
  END IF
  max = 19: counter = 0
  RETURN

pause:
  pause$ = INKEY$: IF pause$ = "" THEN RETURN
  WHILE INKEY$ = "": WEND
  RETURN

listvartable:
  PRINT: PRINT "VARIABLE NAMES TABLE": PRINT
  IF g$ = "P" THEN LPRINT: LPRINT "VARIABLE NAMES TABLE": LPRINT
  byte$ = INPUT$(1,#1): bytecntr = bytecntr + 1
  IF bytecntr/2 = INT(bytecntr/2) THEN byte$ = INPUT$(1,#1)
  WHILE NOT EOF(1)
    byte$ = INPUT$(1,#1): varsize = ASC(byte$)
    varname$ = INPUT$(varsize,#1)
    IF g$ = "P" THEN
      LPRINT USING dc$; varname$;
    ELSE
      PRINT USING dc$; varname$;
    END IF
  WEND
  CLOSE #1
  END

error.msg:
  CLS: PRINT
  IF ERR = 53 THEN
    PRINT "That file not found."
  ELSE
    PRINT "Error number";ERR
  END IF
  END
```

Listing 2

```
REM Compactor
REM By Steve Michel

REM  get file name to compact

  CLS: PRINT
  INPUT "Enter filename to compact";filename.in$
  PRINT: INPUT "Enter filename for compacted file";
filename.out$
  OPEN "I", #1, filename.in$, 1024
  OPEN "O", #2, filename.out$, 1024
  CLS: PRINT: PRINT "Now reading line  => "
  PRINT:  PRINT "Now writing line  => "
  DIM byte$(300)
  lines.in = 0: lines.out = 0: bytes.in = 0: bytes.out = 0
REM directly copy file attribute bytes
  a$ = INPUT$ (1,#1): PRINT #2,a$;
  a$ = INPUT$ (1,#1): PRINT #2,a$;
REM start main read / write loop
loop:
  byte$(1)  = INPUT$ (1,#1)
  linelength = ASC(byte$(1))
  IF linelength = 0 THEN end.of.basic
REM read in line from input file
  bytes.in = bytes.in + linelength
  lines.in = lines.in + 1
  LOCATE 2,22: PRINT lines.in
  FOR J = 2 TO linelength
    byte$(J) = INPUT$ (1,#1)
  NEXT J

REM check for blank line

  byte3   = ASC(byte$(3))
  byte4   = ASC(byte$(4))
  IF byte3 = 0 AND byte4 = 0 THEN loop

REM check for leading apostrophe

  IF byte3 = 58 AND byte4 = 7*25 THEN loop

REM scan current line for imbedded REMs and '

  newlength = 0
  FOR J = 3 TO linelength
    IF byte$(J) = CHR$(174+1) THEN
       newlength = J
       J = 1E+09
    END IF
  NEXT J

  IF newlength = 0 THEN setup.line

REM embedded REM found, check for colon in front of it

  FOR J = newlength TO 3 STEP -1
    IF byte$(J) = CHR$(58) THEN
       linelength = J + 1
       GOTO setup.line
    END IF
  NEXT J

GOTO loop
REM this routine sets up the line length, indentation and
REM two zero bytes at the end of the compacted line
setup.line:
  byte$(1) = CHR$(linelength)
  byte$(2) = CHR$(0)
  byte$(linelength)    = CHR$(0)
  byte$(linelength-1) = CHR$(0)
  bytes.out = bytes.out + linelength
  lines.out = lines.out + 1
  LOCATE 4,22: PRINT lines.out
  FOR J = 1 TO linelength
    PRINT #2, byte$(J);
  NEXT J
  GOTO loop
REM add 2 zero bytes for end of BASIC and check for
REM ODD / EVEN file lengths and adjust if needed
end.of.basic:
  byte$ = INPUT$(1,#1)
  IF bytes.in/2 = INT(bytes.in/2) THEN
    throwaway$ = INPUT$ (1,#1)
  END IF
  PRINT #2,CHR$(0);
  PRINT #2,CHR$(0);
  IF bytes.out/2 = INT(bytes.out/2) THEN
    PRINT #2,CHR$(ASC("J"));
  END IF
REM copy variable table and icon files over
finish.up:
  GOSUB copy.rest
  OPEN "I", #1, filename.in$ + ".info"
  OPEN "O", #2, filename.out$ + ".info"
  GOSUB copy.rest
  KILL filename.out$ + ".info.info"
  LOCATE 6,1: PRINT "All done !"
  END
copy.rest:
  byte$ = INPUT$ (1,#1)
  PRINT #2, byte$;
  IF EOF(1) THEN
    CLOSE #1
    CLOSE #2
    RETURN
  END IF
  GOTO copy.rest
```

•AC•

by John Steiner

Bug Bytes

The Bugs & Upgrades Column

It's late at night and you are just putting the finishing touches on a program. After calling up the compiler, you sit back, relax and wait for the compiler to report on any errors. Out of the corner of your eye, you spot that dreaded, telltale flicker. The power LED is announcing the arrival of your friend and mine, the Guru.

Wondering what is wrong, you calmly reload your program editor and examine your code for the error that prompted the Guru's late night visit. The code is perfect. After hours of trials and many Guru appearances, you decide there must be a bug in the compiler. What should you do? Let others know of the problem before someone else suffers the same fate.

The best way to let others know of the problem is via this column. I plan report all bugs encountered by *Amazing Computing*™ readers. I will go a step further by reporting on upgrades, bug fixes and new releases of Amiga software. This column will be interactive. You document bugs that occur in your software and I will try to verify the existence of the bug you describe and pass it along to the software manufacturer.

The program bug is an elusive creature. He may appear to be in a program, when in fact, he is not. When you come upon a bug, before calling or writing the manufacturer, please:

1. Check, recheck and triple-check the documentation that came with the software.

2. Try to duplicate the bug, noting all actions taken prior to the bug appearance.

3. Note any other windows that are open, how much memory you have left and whether or not there are peripherals involved in the problem. If you are multi-tasking, close off any other tasks and try to duplicate the bug. If the bug only occurs when running other software, things get more complicated. Note the other packages running (the problem could be with them).

4. Make a note of the following information: version or release number of your software, serial number (or other code that identifies you as a legitimate user) of your software, Amiga model number (1000, 500, 2000, etc.), Kickstart and workbench version being used and the amount of RAM in your system. List the type and model of any peripherals that are connected to the Amiga at the time of the problem.

Armed with this information, you are ready to begin tracking down the problem. First, call the dealer who originally sold you the program. In many cases, the dealer has someone on staff who is familiar with the software, or perhaps other customers have run into the problem. The dealer may have already found the solution or perhaps he knows of the availability of a bug fix. It is likely that you will find a previously discovered solution to the problem or it may not really a bug at all. Many times "bugs" are caused by improper operation of the software.

Call the software manufacturers technical support line, detailing the relevant information above and the sequence of events that leads to the problem. Industry insiders report that over 75% of reported "bugs" are solved during the initial call. If the support staff cannot immediately solve your problem, reputable companies will research the problem for you. In the meantime, do some homework. Write down the problem in detail (even better, get the public domain program called Journal and record the steps that occur prior to the bug's occurrence). Mail copies of the problem description, the information you gathered on your system and the Journal playback file to the company and to me. I will try to duplicate your results and notify others via this column.

Readers who have solutions, patches or other knowledge of the problem can report them to me for inclusion in this column. If you have done the leg work on any commercial software packages and found a bug fix, let me know. I will pass along the information to other readers who may be having similar problems.

This column is also an appropriate place for upgrade notices. If you have received an updated version of a software package, especially unpublished updates, please let me know. I will verify if the release is for general distribution and notify readers. Bug fix releases will be published here, so that other readers who are having the same problem can be notified that there is a solution at hand.

For software developers, this column is an appropriate place to announce

continued...

updates, upgrades and bug fixes to your registered users and the general public. Send any listings for this column to:

John Steiner/Bug Bytes
c/o Amazing Computing™
P.O. Box 869
Fall River, MA 02722

Please include a self-addressed stamped envelope if you would like a personal reply.

You may also contact me via People Link or Genie. My address is: Publisher, on both services.

Finally, this month... Electronic Arts has released 1.2 versions of several entries in their popular Amiga software series. Owners of the original versions may upgrade to the new versions by following the procedures listed below.

Upgrade from DeluxePaint to DeluxePaint II or DeluxeVideo to DeluxeVideo II

You can now upgrade to a copy-protected version of either program for $37.00 ($30.00 + 7.00 shipping and handling). You must include the original front cover of your DeluxePaint or DeluxeVideo manual. If you send your non-copy protected version, you will receive all new disks and manuals. To upgrade to a non-copy protected version if your original is copy-protected, enclose an additional $20.00.

Upgrade from DeluxePrint to DeluxePrint 1.2

An upgrade to the copy-protected version is available for $12.00 ($10.00 + 2.00 shipping and handling). You will receive the new 1.2 compatible program disk. If you send your non-copy protected version, Electronic Arts will send along the new 1.2 compatible program disk. To upgrade to a non-copy protected version if your original is copy protected, enclose an additional $20.00.

Upgrade from Instant Music to Instant Music 1.2

An upgrade is available for $12.00 ($10.00 + 2.00 shipping and handling). You will receive the new 1.2 compatible program disk. A non-copy protected version is not currently available.

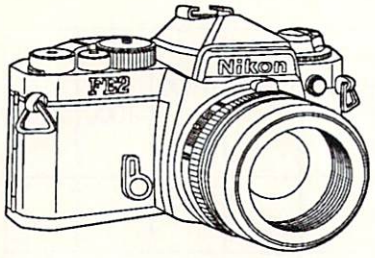
DeluxeMusic Construction Set Update

You can upgrade to improved performance with high end MIDI synthesizers and drum machines, plus higher quality printing for $7.50. Send the last page of the DeluxeMusic manual (labeled "Notice") and you will receive the new DeluxeMusic disk and addenda to the manual. Send all upgrade requests to:

Electronic Arts Amiga Upgrades
Box 7530
San Mateo, CA 94403

They ask only that you send check, money order or Visa/MC credit card info, and allow 4–6 weeks for delivery.

•AC•



Taking the Perfect Screen Shot

by Keith Conforti AC Art Director

Good graphics deserve good screen shots. Let's face it, with 4096 colors and superb resolution, the Amiga™ is an excellent subject for your screen shots (it's also very photogenic). Many people involved with graphics and the Amiga™ don't realize how important and effective good screen shots can be. After all, they appear almost everywhere — magazines, advertisements and software packaging, to name a few.

Required Equipment

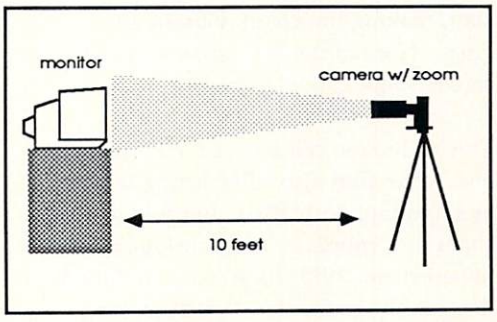
Jim Sachs first filled me in on effectively taking screen shots about six months ago, when Steve Hull interviewed him for Amazing Computing V2.4. As a photographer, I immediately picked up on his valuable tips. Unfortunately, taking screen shots requires some photographic skill and some special tricks of the trade. If you follow these simple guidelines, you'll have no problems at all, though. . . even if you're not a shutterbug.

First, you'll need a 35mm SLR camera. The camera can be manual or automatic, but if you use an automatic, make sure it is equipped with manual override. You'll also need a tripod. I recommend Fuji™ 100 ASA color film because it seems to have more vibrant color for this situation.

Using the standard 50mm lens (supplied with most 35mm SLR cameras) to take your shots will result in a print that looks bowed, or convex, much like the monitor itself. For a more professional look with a flatter image, you must use a zoom lens. The zoom lens flattens the image of the screen, if you take full advantage of its zoom capacity. By setting the tripod approximately ten feet away from the monitor and zooming in on the screen, the convexity is decreased, provided your lens is on the same plane as your monitor (keep them even).

Setting Up Your Shot

The condition under which you take your shots is probably the most crucial aspect. Have you ever seen a perfectly focused screen shot with rich color, marred by glare or a highlight on the screen. The photographer probably took his shots in the wrong environment, with a door or window open, or a monitor or light on nearby? When photographing a dark, reflective surface, leaving lights on in the vicinity is like playing with fire. The room should be absolutely free of direct light. The room need not be pitch black — dim, filtered light is fine, as long as it is angled away from the monitor to avoid glare. The monitor will reflect any light located in front of it, so be sure that any light in the room is placed behind the monitor.



Set the tripod with the camera positioned horizontally. It's very difficult to take a screen shot vertically, unless you are isolating a portion of the screen (or you like to keep your monitor sideways!). Once you have properly prepared the room and loaded the film, you are ready to go. To focus sharply, zoom in totally and focus until the screen and the lines of pixels are sharp, then zoom back to your desired length. A good screen shot leaves about one-half inch of cropping space on all sides of a 3 1/2 by 5 inch print.

Before you snap that shutter, you must make one more important adjustment that may sound a little crazy. Turn down the contrast and the brightness on your Amiga™ monitor just enough to reduce the glare of the bright colors on the screen. If you turn it down too much, the colors of the graphics will be altered and some detail will definitely be lost. Shots taken with normal contrast and brightness are very common and easy to spot. Any bright area in the image will be glowing like a neon billboard.

Snapping the Shutter

Once you've loaded the film, you must take a meter reading in order to know what aperture and exposure settings will produce the best possible shot. The NIkon™ N2000 (the camera I use for screen shots) has a state of the art LED meter. Others have the standard ring and pointer meter in the viewfinder. Both work fine, but the Led is much easier to see if the room and image are dim. Jim Sachs recommends an aperture setting of F-stop 4 and exposure of 1/4 second. Depending on the luminescence of the graphics, you may need a different setting. However, no matter what the settings are, no matter what the subject of your photography is, you should always bracket your shots.

continued...

The Mysteries of Bracketing

To bracket a particular shot, first take a meter reading and set the aperture and exposure (shutter speed). Take one shot at this setting, then increase the shutter speed by one increment, shoot, decrease it by one, shoot, and so on. For example, if the original setting was F-2.8 (aperture) and 1/15 (exposure), your next bracket would be F-2.8 and 1/30 (faster) and 1/8 (slower). You then move to F-4 (smaller aperture), take your bracketed shots, then move to F-1.8 (larger aperture), and so on, until you've bracketed at each aperture setting. As you adjust the aperture settings, you effect the depth of field, but since you're photographing a flat surface, the shots won't be affected. Bracketing insures a successful shot, but uses up quite a bit of film.

SHUTTER SPEED

APERTURE (F-STOP)	1 sec	1/2	1/4	1/8	1/15	1/30	1/60	1/125	1/250	1/500	1/1000	1/2000
1.8					●	●						
2.8				●	●	●						
4			●	●	●	●						
5.6		●	●	●	●							
8		●	●	●								
11	●	●	●									
16												
22												
32												

Figure 1 shows a typical bracketing situation for a single screen shot. A full range consists of six f-stops and six shutter speeds. The most central check under each f-stop would be the meter reading, while the other checks would be your adjustments (two central marks indicate a split meter reading). For a single screen shot, you'd need up to nineteen shots. Just imagine a dozen different shots!

Most exposures will range from 1/30 down. If the rare occasion should arise where the exposure is around 1/250 or faster, beware vertical tracing. The Amiga™ monitor scans sixty times per second, and, if the shutter speed is fast enough, your shot will pick up the scanning resulting in a white line appearing somewhere on the screen. It is similar to the line you'd see if you were photographing a television set.

Developing and Enlarging

Just because you've sweated through your photo session, don't rejoice your success too soon; your responsibility doesn't end there. When you take the film to the photo lab to be developed, or if you develop color yourself, the film should be developed normally. When enlargements are made, be sure to request a higher density, usually positive three or four work best. The increased density produces a darker shot, making the colors vibrant and crisp. Normal density tends to make screen shots look bleached.

This technique can be used with black and white film also. Bracketing is not as important with B&W, but you should shoot at a couple of aperture settings nonetheless. With B&W you can ignore the density instructions when making enlargements.

If you are planning to use slide film, be sure to purchase a slow film speed, probably 64, so you don't run the risk of grainy slides. However, with a slower speed film, you will not have as much of an increase in density (or possibly none at all).

Don't be ashamed if the first couple of rolls are disappointments. It takes a while to memorize the technique and each of the sequences involved. My first roll using this procedure was disastrous because I left a neighboring monitor on. Since then, I've never had a significant problem. Don't be discouraged by early failures. In fact, you should expect them to happen. However, whatever experience you have with a 35mm SLR camera, you'll soon be turning out gallery quality shots. Good luck and happy snapping!

•AC•

Screen Shots: The Good, the Bad and the Ugly

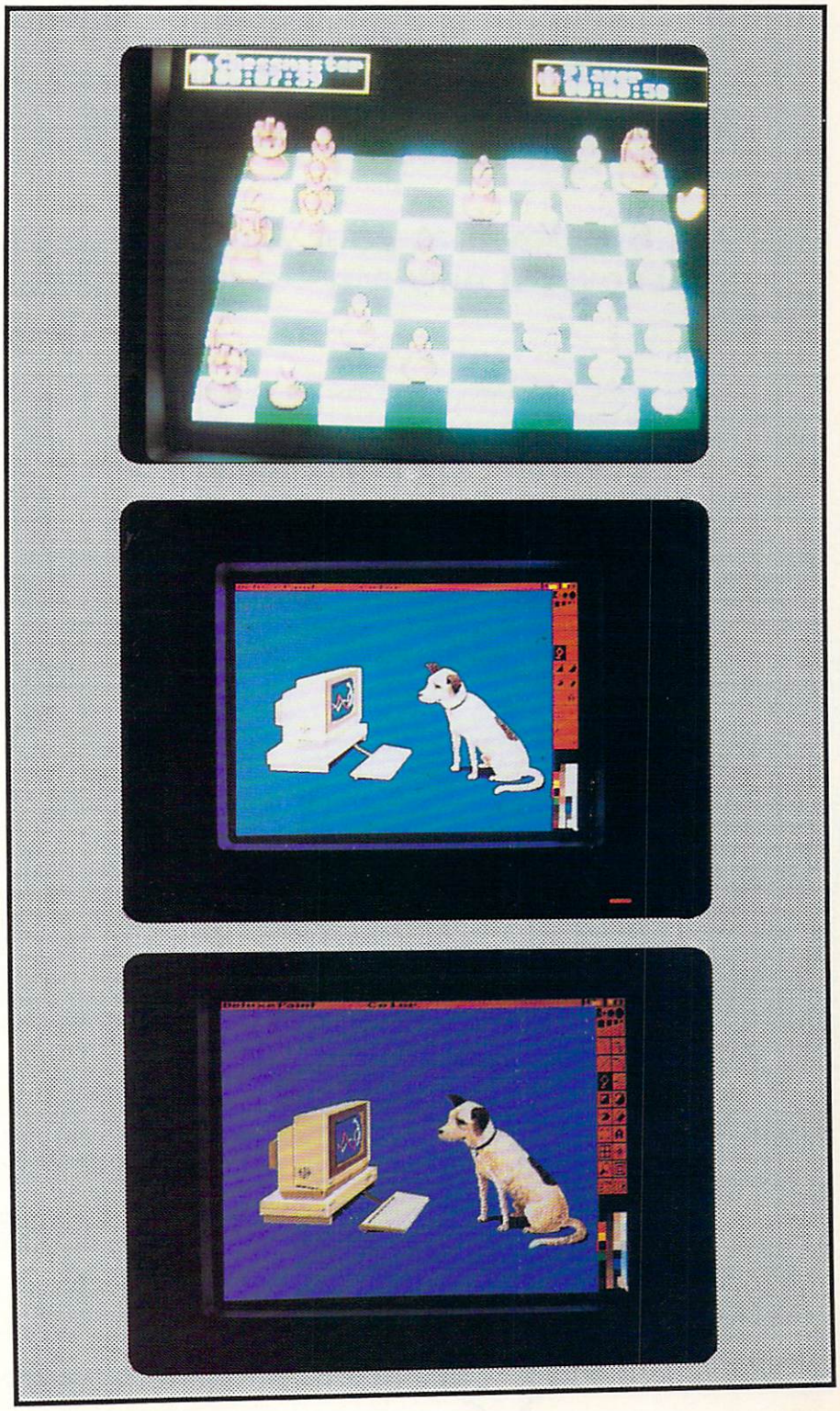
Everyone has seen DaVinci's masterpiece, the Mona Lisa. The essence of her intriguing smile cannot be described in words; it must be seen to be appreciated. The same holds true for any photograph. The reader may get an idea, and possibly understand and visualize all that the writer is trying to describe, but it's just not the same as having a photograph to look at.

The screen shots below were taken before and after I learned the correct procedure. The first set was taken with 100 ASA film and a 50mm lens, without a tripod. I didn't turn down the monitor contrast or brightness. This photo was intended to be a screen shot for Chessmaster 2000™, but it looks more like the glowing neon billboard mentioned earlier. All white areas have a tremendous glow which makes the photo nearly indiscernible. The color is also totally washed out. To top it all off, the shot is not centered correctly because I did not use a tripod, and the great convexity of the screen (flat, when you use a zoom lens) is due to using a 50mm lens from about a foot and a half away.

The second set of photos was taken after I learned Jim Sachs' method. I used a zoom lens and a tripod. I also took precautions against glare by turning the contrast and brightness down and leaving all lights off. The difference is dramatic, when compared to the first shot.

The difference between the two shots of the dog is due to bracketing. You can see the effectiveness of bracketing. This technique offers a wide range of exposures, ensuring a successful shot within your range. Good exposures have more vibrant colors, greater depth and crisp sharpness (as long as your focusing is true).

- Keith Conforti

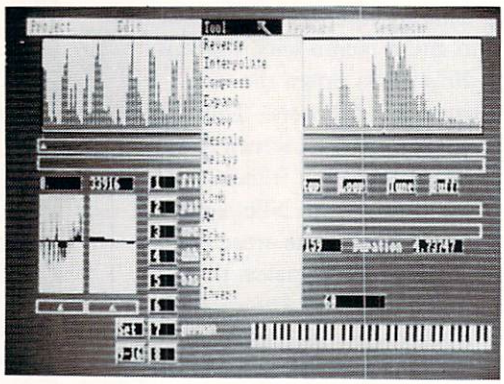


Rediscover Sound

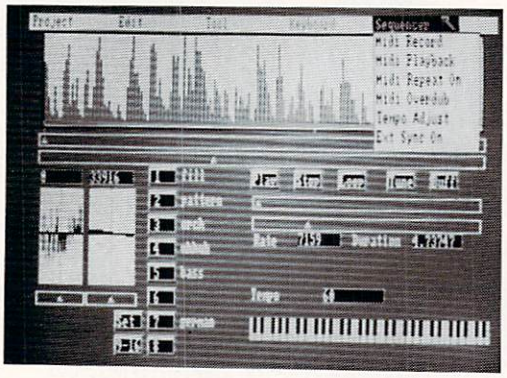
Sit Down at your Amiga with Studio Magic and discover a new way—a better way—to create sound...

Because you'll be working with software modeled after professional digital sound studios.

The Studio Magic tool menu is packed with over a dozen different digital tools. These special effects let you do anything from creating echos, flanges, and delays to enhancing frequency components or performing Fast Fourier Transforms. The Studio Magic tool box puts the power of a digital sound studio at your finger tips. Imagine the possibilities! You can take a voice and make it sound old or young, like Darth Vader, or like an alien from mars. Effects like M-M-Max Headroom become childs play. Your video sound tracks will never be the same!



Studio Magic includes over a dozen digital effects including echos, flanges and delays.



Studio Magic's midi support includes a sequencer with overdub and external sync.

But Studio Magic doesn't only do sound effects... it also does midi. You can record in real time. You can overdub. You can assign any sound to any key on your midi keyboard. Studio Magic even supports advanced midi features such as tempo adjust and external sync. When you combine Studio Magic's midi support with its tools, editing capabilities, and keyboard control, you have the *power* to create spectacular musical pieces. Your amiga never sounded so good.

Studio Magic is IFF compatible. You can import any IFF digital sound, and save in IFF instrument or one-shot format. And of course, Studio Magic works with the Perfect Sound audio digitizer.

Run, don't walk, to your dealer for a demo today!
Suggested retail price **$99.95.**

Studio Magic
SunRize Industries

SunRize Industries
3801 Old College Road
Bryan, Texas 77801
(409) 846-1311

AmigaNotes

by Rick Rae
CIS [76703,4253]

INSIDE MIDI

As it's been about a year since I last talked about MIDI basics, I thought it would be worthwhile to go back and cover the material again. This time, however, we'll go a bit deeper.

MIDI HARDWARE

As pointed out in the earlier introductory article (AmigaNotes, AC V1.7), MIDI stands for Musical Instrument Digital Interface, and is both a hardware and software standard. The hardware is fairly straightforward, so let's get that out of the way first.

MIDI uses a standard serial format of one start bit, eight data bits and one stop bit. This format means that a standard UART, the same sort of chip used to drive an RS232 port, can be used for MIDI. Rather than requiring a special baud rate generator, MIDI operates at a frequency of 31.25 kilobaud, which can usually be derived by dividing the system clock by a power of two.

Rather than using voltage levels, MIDI is based on a five milliamp current loop; a logical one is represented by an absence of current. MIDI outputs are very simple, usually consisting of little more than a resistor-protected current source and a switch, such as a transistor, TTL gate, or op amp. (See AmigaNotes, AC V2.2 for one approach.)

A MIDI input is only slightly more complex. The specification requires that the receiver to be completely isolated to eliminate hum and safety problems. This isolation is usually achieved with an optoisolator, an integrated circuit which combines a light emitting diode with a photosensitive component (usually a transistor). In addition, the MIDI shield pin is left unconnected on MIDI inputs. The switched current from the MIDI transmitter travels down the cable, through the receiver's optoisolator, and back up the cable to the transmitter. In one sense, there is never any electrical connection between two MIDI devices — the data is ultimately received via a beam of light.

MIDI cables have specifications, too. The cable itself should be a shielded, twisted pair, not longer than fifty feet (longer runs have been used successfully, especially with low-capacitance cable). Each end is terminated in a five pin, 180 degree male DIN connector. The twisted pair interconnects pins four and five, while the shield is tied to pin two on both ends.

MIDI PROTOCOL: CHANNEL MESSAGES

MIDI communication is based on messages. A message is composed of a status byte, which is optionally followed by data bytes. This arrangement can be confusing at first, but bear with me and we'll chip it away bit by bit (no pun intended, honest).

Each status byte may be broken down into fields as shown:

T SSS CCCC

The most significant bit identifies what each byte for us — A set means a status byte; a zero means a data byte. So, the first byte of any message, the status byte, will have the message's most significant bit set. It may be followed by data bytes, which will have zeroes in this position.

The next three bits indicate the type of message. Since there are eight possible combinations of three bits, it makes sense that there are eight possible types of status bytes. Let's quickly run down the list.

000: NOTE OFF (3 bytes)

This message tells the synthesizer to turn off a note which (supposedly) was previously turned on. The second byte indicates which of the 127 MIDI notes should be turned off. The third byte, release velocity, indicates how quickly the key was released. This byte is not supported by most current keyboards.

001: NOTE ON (3 bytes)

This message tells the synthesizer to turn on a note. . . usually. The second byte indicates which note is to be turned on. The third byte is velocity, or how quickly the key was struck. Velocity information is often used to control the loudness or brightness of a sound and, while many keyboards do not send this information, quite a few synthesizers will respond to it. Almost all synthesizers respond to a special case — however, a note on message with a velocity of zero is equivalent to a note off. The reason for this reaction will be explained later (trust me).

continued...

010: POLYPHONIC KEY PRESSURE (3 bytes)

This message indicates how hard a key is being depressed once it has reached the stop. The second byte indicates which note is being depressed, while the third byte indicates the amount of pressure applied. By using key pressure to control pitch or timbre, experienced 'synthesists' can put quite a bit of feeling into their music without ever lifting their hands from the keyboard. Unfortunately, polyphonic key pressure requires a separate pressure sensing mechanism under each and every key. Like release velocity, this message is usually sent only by the more expensive devices.

011: CONTROL CHANGE (3 bytes)

This message is sort of "generic"; it means something, somewhere, has changed. The second byte specifies the controller number, the third notes the value. The function of each controller is decided by the individual equipment manufacturer, but a few, more-or-less, standards have emerged, some of which are shown here:

CONTROL CHANGE MESSAGE

Byte 2	Description of Controller
1	Oscillator Modulation
2	Timbre Modulation (Filter)
7	Volume
64	Sustain
65	Portamento

Provisions have been made for quite a few controllers. Values of 0 through 31 are reserved for continuous controls like knobs, sliders, etc. Each of these controls can take on any one of 128 values. If more resolution is required by a particular controller, controller numbers 32 through 63 represent seven extra bits of resolution for the first 32 controllers.

In other words, a control change message with a second byte of 0 would provide the seven most significant bits for the first controller. A subsequent control change message with a second byte of 32 would provide the seven least significant bits for the same controller, resulting in a total of 14 bits of resolution. This approach was chosen because most continuous controls don't need more than seven bits of resolution, and always adding an additional byte would degrade performance.

If the second byte falls in the range of 64 to 95, the controller is assumed to be a switch, and the third byte will simply indicate whether the switch is on or off.

Values 96 through 121 are currently undefined, but may be assigned at a later date.

Values above 121 are reserved for special channel mode messages as follow:

CONTROL CHANGE MESSAGE

Byte 2	Message Type
122	Local Control
123	All Notes Off
124	Omni Mode Off
125	Omni Mode On
126	Mono Mode On
127	Poly Mode On

Local Control determines whether the synthesizer's keyboard controls the internal sound generators or merely generates MIDI output. All Notes Off is a sort of "panic button"; it is equivalent to sending a note off for every note. The last four messages determine the synthesizer's channel mode, which we'll touch on in just a moment.

100: PROGRAM CHANGE (2 bytes)

This message initiates a program ("patch") change. The data byte is the program number selected.

101: MONOPHONIC PRESSURE (2 bytes)

This message (also called Channel Pressure or AfterTouch) is similar in function to Polyphonic Key Pressure, but generates only one pressure value across an entire keyboard. The data byte represents the applied pressure.

110: PITCH WHEEL CHANGE (3 bytes)

Since the pitch wheel is one of the few controllers which may easily require more than seven bits of resolution, it was given its own status message. The second byte holds the seven least significant bits; the third byte holds the seven most significant.

All these change messages are classified as channel messages. MIDI defines sixteen data channels, allowing one cable to carry information for several synthesizers or voices. These messages use the last four bits of the status byte to specify the desired channel. How each synthesizer responds to channel messages is determined by its channel mode, which has four possible states:

OMNI ON POLY:

This state is often the default power-up mode for any synthesizer which does not have non-volatile memory. Omni on means the synthesizer will respond to any message on any channel. This arrangement is handy for initial setup or quick tests. In combination with this full-ranging response, poly means the synthesizer will play as many notes as possible — if it is an eight voice machine, you can play an eight note chord.

OMNI ON MONO:

This mode means the synthesizer will respond to all messages regardless of channel, but will only play one note at a time. This mode is probably the least used of the four. Unless you have some special considerations such as an old analog two-voice synthesizer, you won't be using this one very much.

OMNI OFF POLY:

This setting is the most popular and causes the synthesizer to respond polyphonically, but only to messages on one channel. In this way, many devices can share one MIDI connection. On some synthesizers, the transmit channel is preset and not changeable, but most machines do allow you to set the receive channel.

continued...

ATTENTION! DELUXE MUSIC CONSTRUCTION SET™ USERS

ATTENTION! MUSIC STUDIO™ USERS

SYMPHONY SONGS

ATTENTION! INSTANT MUSIC™ USERS

SYMPHONY SONGS gives you a library of nearly 1,000 music masterpieces. All songs are in IFF format so they may be loaded, played, printed, transposed, and modified in any way you like using your favorite composition program. Included is a free program to convert the IFF files to MUSIC STUDIO™ format.

The songs have been arranged by C. Clark Rulaford and Randy Spector and take advantage of the full **4 Voice** capability of the AMIGA.

Space does not allow listing all the songs in each of the volumes. We have listed a few and show the total number in each volume as well as the playing time. A complete list of songs may be purchased for **$3.95.** Each volume of the 27 volumes listed is **$24.95.**

BEATLES Part 1
Vol 15 (21 Pieces 40 Min)
Let It Be, Yesterday, Eleanor Rigby, When I'm 64, . . .

BEATLES Part 2
Vol 40 (17 Pieces 40 Min)
Magical Mystery Tour, Lucy In The Sky With Diamonds, Penny Lane, . . .

CLASSICAL Part 1
Vol 27 (19 Pieces 40 Min)
Prelude #1, Moonlight Sonata 1st and 2nd Movement, . . .

CLASSICAL Part 2
Vol 34 (15 Pieces 40 Min)
Sonata In C Major, Jesus Joy Of Man's Desire, . . .

CLASSICAL Part 3
Vol 31 (18 Pieces 35 Min)
1st Piano Concerto, Polonaise Sonata In C Major, Etude #3, . . .

CLASSICAL Part 4 (Bach)
Vol 35 (22 pieces 30 Min)
Two Part Invention #1, Three Part Invention #6, Prelude and Fugue 1, . . .

CLASSICAL Part 5 (Bach/Clementi)
Vol 46 (24 Pieces 50 Min)
Choral #1, Sonata #1, Theme and 11 Variations From The 2nd Sonata, . . .

BEETHOVEN, BROADWAY, & BLUES
Vol 38 (15 Pieces 40 Min)
2nd Movement Of the Pathetique Sonata, Minuet In G, Fuer Elise, . . .

COUNTRY CLASSICS Part 1
Vol 41 (18 Pieces 45 Min)
Thank God I'm a Country Boy, Act Naturally, . . .

ROCK Part 1
Vol 32 (19 Pieces 50 Min)
AXEL F, Eye Of The Tiger, Both Sides Now, . . .

ROCK Part 2
Vol 16 (21 Pieces 40 Min)
Georgy Girl, Guantanamera, Theme From "Love Story," Cherish, . . .

80's GREATEST
Vol 24 (16 Pieces 50 Min)
Hill Street Blues Theme, Chariots Of Fire Theme, Dynasty Theme, . . .

70's GREATEST
Vol 12 (21 Pieces 45 Min)
Tie A Yellow Ribbon On The Old Oak Tree, We've Only Just Begun, . . .

60's GREATEST
Vol 13 (21 Pieces 45 Min)
Windy, By The Time I Get To Phoenix, Come Saturday Morning, . . .

GOLD & PLATINUM HITS
Vol 45 (19 Pieces 60 Min)
Thriller, 99 Luft Balloons, California Girls, . . .

KENNY RODGERS HITS
Vol 39 (12 Pieces 45 Min)
Lady, Ruby, She Believes In Me, The Gambler, . . .

BILLY JOEL GREATEST HITS
Vol 43 (17 Pieces 65 Min)
Piano Man, Say Goodbye To Hollywood, Only The Good Die Young, . . .

COUNTRY CLASSICS Part 2
Vol 42 (19 Pieces 50 Min)
Ode To Billy Joe, Me and Bobby McGee, Country Roads, . . .

TV THEMES
Vol 37 (21 Pieces 35 Min)
Hill Street Blues, St. Elsewhere Theme, Masterpiece Theater Theme, . . .

MOVIE THEMES
Vol 19 (23 Pieces 40 Min)
MASH Theme, The Rose, Can You Read My Mind (Superman), . . .

BROADWAY'S THEMES
Vol 47 (25 Pieces 65 Min)
The Last Supper, Dr. Doolittle, The Old Dope Peddler, . . .

CHURCH MUSIC
Vol 28 (26 Piece 50 Min)
Amazing Grace, What A Friend We Have In Jesus, . . .

BARBERSHOP
Vol 22 (22 Pieces 45 Min)
Hello Dolly, Put On a Happy Face, Hey Look Me Over, . . .

RICHARD RODGERS SONGBOOK
Vol 18 (19 Pieces 40 Min)
Climb Every Mountain, DO-RE-MI, The Sound Of Music, . . .

NOSTALGIA
Vol 17 (22 Pieces 45 Min)
Let Me Call You Sweetheart, Ain't Misbehavin', On The Goodship Lollipop, . . .

CHRISTMAS
Vol 36 (26 pieces 50 Min)
O Little Town Of Bethlehem, Let It Snow, March Of The Toys, . . .

POLKA PARTY
Vol 33 (18 Pieces 40 Min)
Happy Polka, Pizzacato Polka, Betty Polka, . . .

SYMPHONY JUKEBOX

Symphony Jukebox allows you to program a selection of songs, their order as well as the number of times, and allows you to listen to them for hours of uninterrupted playing. Other features include MIDI output, instrument selection, transposition, and tempo modification. **$24.95**

SYMPHONY MUSIC VIDEO

This program has all the features of our SYMPHONY JUKEBOX, however, it also allows you to specify a picture to be displayed with each song. The pictures and music are all in standard IFF format so you may use the songs and pictures included, or use those you developed with your music program (i.e. DMCS), or your paint program (i.e. Deluxe Paint). Included are Christmas music and pictures. **$24.95**

We accept CASH, CHECK, COD, VISA and MASTER CARD orders.
Shipping and handling US and Canada $3.00
Shipping and handling outside the US and Canada $5.00
COD charge $2.00
Illinois residents add 6¼% sales tax.

VISA MasterCard

Dealer Inquires Invited

Speech Systems
38W255 DEERPATH ROAD
BATAVIA, ILLINOIS 60510
(312) 879-6811

Deluxe Music Construction Set, Deluxe Paint, Instant Music are trademarks of Electronic Arts. Music Studio is a trademark of Activision.

OMNI OFF MONO:

This mode is gaining popularity as synthesizers become more powerful because it allows multi-timbral operation. In other words, by using this mode, your synthesizer can play several different patches at once. Here's how it works: since omni is off, the machine will only respond to one channel, called the basic channel. Since mono mode is selected, only one voice will sound. However, the next higher voice will respond in the same way on the basic channel plus one, the next voice on the next channel, and so on. Voice one could be a bass guitar, voice two a lead guitar, voices three through six an acoustic piano . . . in other words, you can simulate an entire band with one instrument.

Two notes about this mode — first, it's interesting to note that it would be extremely difficult, if not impossible, to simultaneously play various patches without using a sequencer, so, how would you play four different patches on one keyboard? About the best you could do would be to split the keyboard into zones and assign one instrument sound to each zone. With a sequencer running the show, this problem goes away, since each instrument sound would have its own track.

Second, many synthesizers do not support this mode (some support it in a limited fashion). If you're considering buying a synthesizer and have certain things you want to do with it, be sure to ask before plunking down your money.

Astute readers will notice that I've only listed seven of the eight possible status bytes. The eighth code, 111, is reserved for...

SYSTEM MESSAGES

A system message is intended as a sort of global announcement, so that no channel numbers are used. Instead, the four least significant bits are used to indicate the type of system message. It follows, then, that there are 16 possible system messages:

SYSTEM COMMON MESSAGES

Status Byte	Bytes	Message Type
1 111 0010	3	Song Position
1 111 0011	2	Song Select
1 111 0110	1	Tune Request

SYSTEM REAL-TIME MESSAGES

Status Byte	Bytes	Message Type
1 111 1000	1	Timing Clock
1 111 1010	1	Start
1 111 1011	1	Continue
1 111 1100	1	Stop
1 111 1110	1	Active Sensing
1 111 1111	1	System Reset

SYSTEM EXCLUSIVE MESSAGES

eg: save

Status Byte	Bytes	Message Type
1 111 0000	>=2	System Exclusive
1 111 0111	1	End System Exclusive

As shown, system messages can be grouped into three categories. System Common messages deal with non-timed functions and are intended to be received by all synthesizers in a system. System Real-Time messages deal with synchronization of the devices and are also intended for all synthesizers in a system. System Exclusive messages generally deal with bulk patch and sample dumps and remote programming or storage and are intended for all synthesizers by a particular manufacturer. Again, let's run down the list.

Song position is the number of MIDI beats which have passed since the start of a song. When a sequence is started, this value is set to zero and begins incrementing. By using the Song Position Pointer message to preset the pointer, you can start recording or playback in the middle of a song. This feature is most useful when dealing with drum machines or multiple sequencers.

Most drum machines and sequencers can store more than one pattern or song. When using these devices in a MIDI system, a Song Select message may be used to specify which sequence is to be played.

The newer 'synthesists' among us have been spared the frustration of the older analog synthesizers, which had a tendency to drift with changes in temperature and line voltage. At the very least, these machines had to be tuned when powered up. Some of the more sophisticated analog machines included circuitry which allowed them to tune themselves. A Tune Request message instructs these instruments to do the tuning.

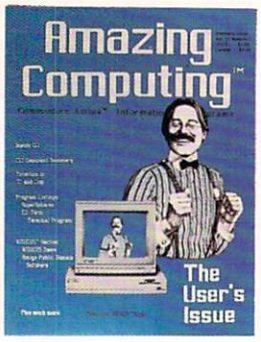
The Start, Continue, Stop and Timing Clock messages are involved with real time recording or playback. Their functions should be fairly obvious.

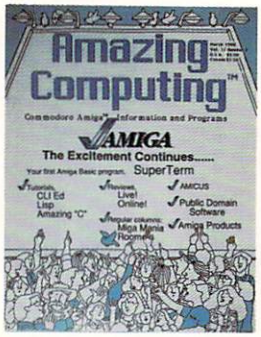
Active Sensing is an optional MIDI feature. If an Active Sensing message is sent to a synthesizer, the instrument will expect to continue receiving Active Sensing messages at regular intervals. If all MIDI messages are interrupted for more than a few hundred milliseconds, the synthesizer will assume there is a problem and turn off all active notes. This feature can, at least, help reduce the embarrassment when you trip over a MIDI cord on stage.

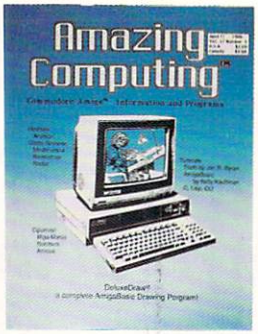
To be honest, I've never understood why System Reset was included among the real-time messages, but there it is. A System Reset message initializes a synthesizer to its power-up condition. The MIDI specification warns that this command should be "used sparingly." Think of it as a three finger reset on your Amiga.

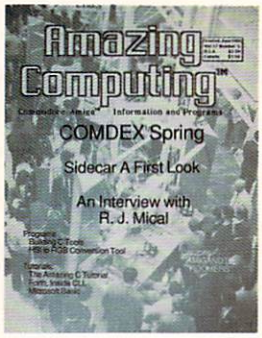
System Real-Time messages are the only status bytes which start with the code 1 111 1. Because of this distinguishing detail, such messages can be easily recognized. This easy recognition is fortunate — since Real-Time messages are time critical, they might arrive at

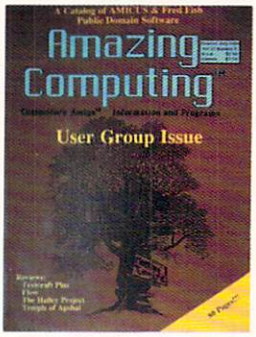
continued...

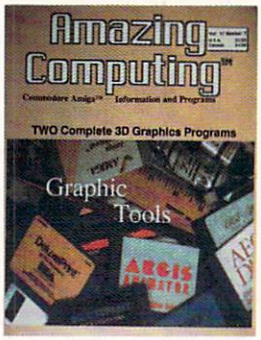


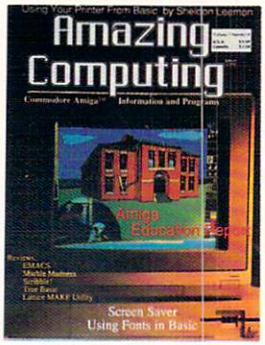


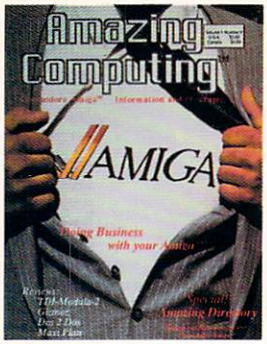


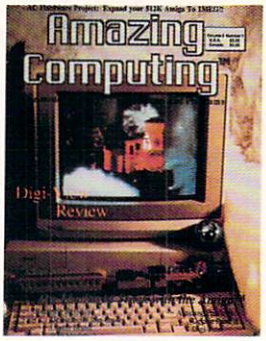


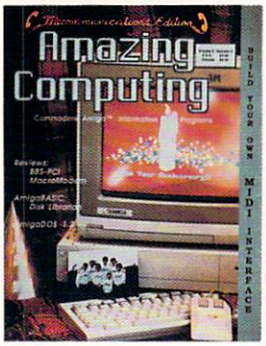


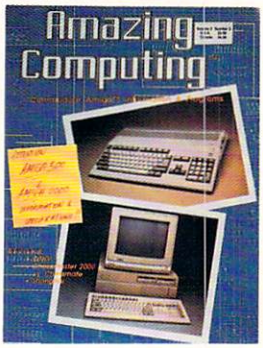


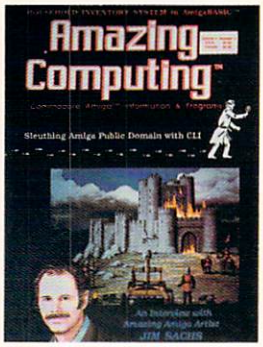


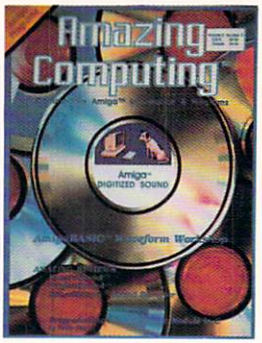


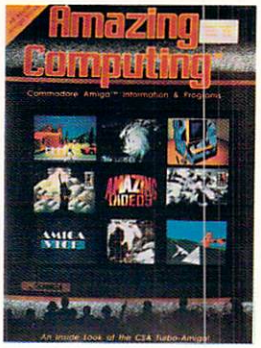


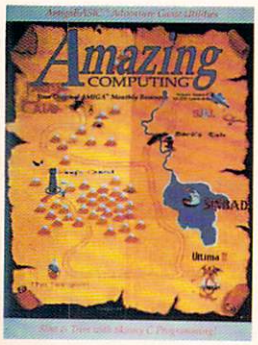


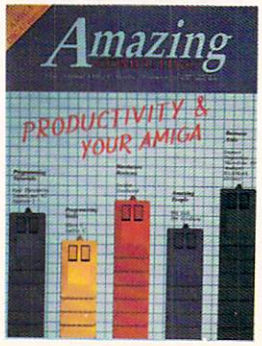












To be continued.....

any time . . . even within other messages. A MIDI device must be intelligent enough to pick the real time messages out of the data stream and deal with them separately, while not allowing them to disturb the other information.

The System Exclusive messages are sort of loopholes, through which an equipment manufacturer can push just about anything imaginable. A System Exclusive message (abbreviated SOX, Start Of eXclusive) signals all synthesizers that a special message is about to begin. The second byte is a manufacturer's identification code. All MIDI manufacturers who wish to use system exclusive information must register with the proper authorities and be assigned an ID.

When a synthesizer receives a system exclusive message, it checks the second byte to see which brand of equipment the message is intended for. If no match is found, the entire message is ignored. If the message is for the brand of synthesizer in use, the instrument uses the data from that point to the End System Exclusive (EOX, End Of eXclusive) message. The format of the data between these two messages, and what the synthesizer does with that data, is totally undefined and left to the manufacturer's discretion.

REDUCING OVERHEAD

In the hardware discussion, I mentioned that the MIDI data rate is 31.25 Kbaud (31,250 bits per second). I also mentioned that a MIDI byte was eight bits, bracketed by a start bit and a stop bit. A little math shows that it takes 320 microseconds to send one MIDI byte. Recall that Note On and Note Off messages require three bytes each, which means it takes 960 microseconds – call it one millisecond – to turn a note on or off. A single seventh chord over a bass note will take at least five milliseconds to transmit, with another five milliseconds needed when the keys are released. It is obvious that, with a lot of activity, a MIDI data stream could simply run out of bandwidth.

MIDI tries to ease this problem with running status. The MIDI specification states that, once a channel message status byte is received, the receiver should remain in that mode, until another status byte is received. In other words, if a synthesizer is sent a Note On message, it expects all further data to be Note On data, until another message is received. Rather than sending three bytes for each Note On message, we need only send the two data bytes from each message.

This method works wonders for playing a chord, but when the first key is released, the Note Off message would interrupt the flow – which is why a zero velocity Note On is equivalent to a Note Off. By sending zero velocity Note On data, instead of a Note Off command, running status can be much more efficient. If most of the activity on a MIDI port is simply notes beginning and ending, running status can reduce overhead by almost 33%.

RESOURCES

That pretty well covers the technical aspects of the MIDI specification. If you'd like to see the "real thing," you can order the MIDI spec from the International MIDI Association . . . or, better yet, pick up a copy of Craig Anderton's MIDI for Musicians. This 100-plus page book explains MIDI better than I ever could in the space I have available here. It also includes a reprint of the abbreviated MIDI specification.

If you're going out to buy a synthesizer, play the keyboards and ask questions! Near the back, you'll usually find a MIDI Implementation Chart, which will tell you exactly what the synthesizer is capable of when being used as a MIDI instrument.

Until next month... Nybbles, Rick

For more information, contact:
International MIDI Association
11857 Hartsook Street
North Hollywood, CA 91607
(818) 505–8964

•AC•

Directory Listings Under AmigaDOS™

by Dave Haynie

Part II — More Explanations and Listing

I've come up with many ideas to help speed up the directory commands at various levels. The ASDG company plans to introduce an intelligent hard disk controller that actually implements a sorting and look ahead scheme in its firmware, combined with caching. This hard drive could be very fast; it would certainly be faster than the FDir command run on a hard drive, and with the sorting taking place at the handler and/or device level, the speedup would be all-around, not just embedded in a directory command.

On the subject of directory commands, Leo L. Schwab, of display hack fame, has been developing something very similar to the ASDG drive controller independently (his version is called "eless"; he recently released it to Usenet). Just today, Paul Higginbottom, of Commodore Sales, came to me asking what I was doing with this idea. He'd been playing with some of the same concepts . . . obviously, this idea's time has come.

```
/* ------------------------------------------------------------------
*/

/*              Fast Directory Command  by Dave Haynie

   This program is an example of DOS Packet calls, and of the described
   method of reading a standard AmigaDOS directory that's a bit faster
than
   regular directory programs, which rely on the ExNext() call or the
   ACTION_EXAMINE_NEXT packet.  The fast method is device dependent; a
   standard, equivalent directory function is available as a fallback for
   other file handlers, and for comparison purposes (via the -n option).
*/

#include <exec/types.h>
#include <exec/memory.h>
#include <exec/exec.h>
#include <libraries/dos.h>
#include <libraries/dosextens.h>
#include <libraries/filehandler.h>
#include <ctype.h>
#include <stdio.h>

/* ------------------------------------------------------------------
*/

/* Program's text messages */

#define NO_FILES        "No Files Found\n"
#define ERR_DIR         "Error: Directory not found\n"
#define NOT_DIRECTORY   "Error: Not a directory\n"
#define OLD_LIBRARY     "Error: DOS Library too old\n"

/* Important, fixed, block numbers */

#define HASHSIZ         72L             /* Hash table size */

/* Standard output file pointer */

BPTR Stdout = NULL;

struct DosLibrary *DosBase = NULL;

/* Standard DOS FileInfo stuff, for various EXAMINE packets */

typedef struct FileInfoBlock FIB;

#define FIBMEM          (MEMF_PUBLIC|MEMF_CLEAR)

/* ------------------------------------------------------------------
*/

/* This section contains the code that maintains a sorted list of directory
   keys.  When a directory block is first read, the key list will be built
   with directory keys from the hash table.  Each key processed in the main
   directory loop may in turn have a hash table collision key of its own
   for insertion in the list.  As keys are processed, they are removed from
   the front of the list, so there can only be a maximum of "HASHSIZ" keys
   in the list at any given time.  Variable "dir" indicates the direction
   in which the list is being built, while "top" points to the top of the
   sorted list. */

#define DOWN    0       /* Heads are moving down */
#define UP      1       /* Heads are moving up */

short dir = UP;         /* The current movement of the heads */

unsigned top = 0;       /* Indexes the top of the keys list */
LONG keys[HASHSIZ+1];   /* The actual keys list */

/* This macro pops the key from the top of the keys[] list. */

#define PopKey(k)   { \
   dir = (((k) = keys[0]) > keys[1]) ? DOWN : UP; \
   movmem((char *)&keys[1],(char *)&keys[0],(-top)<<2); }

/* This function adds the given key "keyno" into the key list.  This list
   will always be built in a sorted order, based on "dir" and the "guide"
   parameter.  When a directory header is processed, the keys from the hash
   table are inserted in ascending order, based on a direction of "UP".
   This means keys larger than "guide" are put in ascending order at the
   head of the list, keys smaller are put at the tail of the list in
   descending order.  The process reverses if "dir" is "DOWN".  */

void AddKey(keyno,guide)
register LONG keyno, guide;
{
   register short i;

   if (dir == UP && guide < keyno)
      for (i = 0; i < top && keys[i] < keyno; i++);
   else if (dir == DOWN && guide > keyno)
      for (i = 0; i < top && keys[i] > keyno; i++);
   else if (dir == UP)
      for (i = top; i > 1 && keys[i-1] < keyno; i--);
   else
      for (i = top; i > 1 && keys[i-1] > keyno; i--);
   movmem((char *)&keys[i],(char *)&keys[i+1],(top++-i)<<2);
   keys[i] = keyno;
}

/* ------------------------------------------------------------------
*/

/* This section contains the device-specific cylinder functions. */

short CYLINDERSIZ;              /* Size of cylinder or track */

BOOL trackonly = TRUE;          /* Check single tracks, not whole cylinder
*/

/* This macro compares two key values.  It returns TRUE if they're on the
   same cylinder (any head), FALSE otherwise. */

#define CylEqu(k1,k2)   ((k1)/CYLINDERSIZ == (k2)/CYLINDERSIZ)

/* This function looks up the device in use in the device list, based on
   the volume's handler task.  From this, the cylinder size is calculated
   for use by the same-cylinder test.  This could be easily changed to
   lookup only same-track info if desired.  This function returns TRUE if
   the device looks OK, FALSE if it seems to have an unusual file
   structure. */
```

continued...

```
BOOL GetCyl(dl)
struct DeviceList *dl;
{
   register struct DeviceNode *dn;
   struct RootNode *root;
   struct DosInfo *info;
   struct FileSysStartupMsg *fssm;
   LONG *env;

   root = (struct RootNode *) DosBase->dl_Root;
   info = (struct DosInfo *) BADDR(root->rn_Info);
   dn = (struct DeviceNode *) BADDR(info->di_DevInfo);

   for (; dn != NULL; dn = (struct DeviceNode *)BADDR(dn->dn_Next))
      if (dn->dn_Type == DLT_DEVICE && dl->dl_Task == dn->dn_Task) {
         fssm = (struct FileSysStartupMsg *) BADDR(dn->dn_Startup);
         env = (LONG *) BADDR(fssm->fssm_Environ);
         if (env[DE_SIZEBLOCK] != 128L) return FALSE;
         CYLINDERSIZ = env[DE_BLKSPERTRACK]
               * (trackonly?1L:env[DE_NUMHEADS]);
         return TRUE;
      }
   return FALSE;
}

/* ------------------------------------------------------------------
*/

/* This section contains my implementation of typed DOS Packet functions,
   based on the BigPacket type. */

/* This structure stores all of the information required in a DOS packet
   including a pointer to the reply port that's used to get the results
   of a packet action and data pointer, which may point to a block or other
   data object, based on the specific packet type allocated.  */

typedef struct {
   struct Message       bp_Msg;
   struct DosPacket     bp_Pkt;
   struct MsgPort       *bp_Rep;
   CPTR                 bp_Data;
} BigPacket;

#define PACMEM          (MEMF_PUBLIC|MEMF_CLEAR)

#define DOSTRUE         -1L
#define DOSFALSE        0L

/* This macro returns TRUE if the given packet has been sent, FALSE
   otherwise. */

#define WasSent(pkt)    ((pkt)->bp_Rep != NULL)

/* This function allocates a typed BigPacket.  It returns NULL if the
   packet can't be allocated for some reason.  The allocation is not
   completely general, but it works fine for the packets we're concerned
   with in this example. */

BigPacket *AllocPacket(action,size,memory)
LONG action,size,memory;
{
   BigPacket *pkt;              /* Resulting packet */

   if ((pkt = (BigPacket *) AllocMem(sizeof(BigPacket),PACMEM)) == NULL)
      return NULL;
   pkt->bp_Pkt.dp_Action = action;
   if (size != 0L) {
      if ((pkt->bp_Data = (CPTR) AllocMem(size,memory)) == NULL) {
         FreeMem(pkt,sizeof(BigPacket));
         return NULL;
      }
      pkt->bp_Pkt.dp_Arg2 = ((ULONG)pkt->bp_Data) >> 2;
   }
   pkt->bp_Msg.mn_Node.ln_Name = (char *) &(pkt->bp_Pkt);
   pkt->bp_Pkt.dp_Link         = &(pkt->bp_Msg);
   return pkt;
}

/* This function waits for the given packet of any kind to complete its
   action, returning its first return status value. */

LONG WaitPacket(pkt)
BigPacket *pkt;
{
   if (!WasSent(pkt)) return DOSFALSE;
   WaitPort(pkt->bp_Rep);
   GetMsg(pkt->bp_Rep);
   DeletePort(pkt->bp_Rep);
   pkt->bp_Rep = NULL;
   return pkt->bp_Pkt.dp_Status;
}

/* This function asynchronously sends any packet "pkt" to perform its
   function.  It is assumed the particular packet-type-specific data has
   been properly initialized.  If the packet has already been sent, the
   function will return DOSFALSE. */

LONG SendPacket(pid,pkt)
struct MsgPort *pid;
BigPacket *pkt;
{
   extern struct MsgPort *CreatePort();

   if (WasSent(pkt)) return DOSFALSE;
   if ((pkt->bp_Rep = CreatePort(NULL,0L)) == NULL) return DOSFALSE;
   pkt->bp_Pkt.dp_Port = pkt->bp_Rep;
   PutMsg(pid,pkt);
   return DOSTRUE;
}

/* This function frees up a packet allocated with AllocPacket().  If the
   packet was asynchronously sent, and has not yet returned, the function
   will WaitPacket() for the action to complete. */

void FreePacket(pkt,size)
BigPacket *pkt;
LONG size;
{
   if (WasSent(pkt)) WaitPacket(pkt);
   if (pkt->bp_Data != NULL)
      FreeMem(pkt->bp_Data,size);
   if (pkt->bp_Rep != NULL) DeletePort(pkt->bp_Rep);
   FreeMem(pkt,sizeof(BigPacket));
}

/* This function is a general synchronous packet routine.  It queues the
   given packet, then waits for it to complete before returning. */

LONG DoPacket(pid,pkt)
struct MsgPort *pid;
struct BigPacket *pkt;
{
   if (SendPacket(pid,pkt) != DOSTRUE) return DOSFALSE;
   return WaitPacket(pkt);
}

/* ------------------------------------------------------------------
*/
```

```
/* This section contains my implementation of the standard file handler's
   BLOCK and related function and other data, for use with the packet
   ACTION_GET_BLOCK. */

/* Variable sized BCPL type character string, in easy to use C terms.
   The sizing parameter is in terms of long words. */

#define VARBSTR(s) struct { UBYTE len; BYTE str[(s<<2)-1]; }

/* File block typing information */

#define FT_SHORT_FILE   2L          /* File or Directory block */
#define FT_DATA_BLOCK   8L          /* Data block */
#define ST_FILE         0xfffffffd  /* File subtype */
#define ST_ROOT_DIR     1L          /* Root directory subtype */
#define ST_USER_DIR     2L          /* User directory subtype */

/* The is the DOS block definition.  The interesting fields for most
   directory items are called out in this structure.  Only a few are
   actually used in this example, the rest of them could be added to an
   extended version of this directory command. */

typedef struct BLOCK {
   LONG                 b_type;             /* Block's primary type
*/
   LONG                 b_key;              /* Pointer back to self
*/
   LONG                 b_seq;              /* Blocks used */
   LONG                 b_datasize;         /* Data blocks used */
   LONG                 b_first;            /* File's first datablock
*/
   LONG                 b_checksum;         /* Block checksum */
   LONG                 b_hash[HASHSIZ];    /* Directory Hashtable */
   LONG                 b_spare[2];         /* 2 Spares */
   LONG                 b_protect;          /* Protection */
   LONG                 b_bytesize;         /* File size, in bytes */
   VARBSTR(23)          b_comment;          /* Filecomment */
   struct DateStamp     b_date;             /* Creation date */
   VARBSTR(16)          b_name;             /* Filename */
   LONG                 b_collis;           /* Hash collision chain
*/
   LONG                 b_parent;           /* Parent directory */
   LONG                 b_ext;              /* File extension block
*/
   LONG                 b_sub;              /* Block Subtype */
} BLOCK;

#define BLKMEM          (MEMF_CHIP|MEMF_PUBLIC)

/* This function does essentially the same thing as DoPacket(), but
   specifically for a block read with an initialized ACTION_GET_BLOCK
   packet.  The sector to read from is also supplied. */

LONG ReadBlk(pid,pkt,sect)
struct MsgPort *pid;
BigPacket *pkt;
LONG sect;
{
   if (sect == 0 || pkt->bp_Pkt.dp_Action != ACTION_GET_BLOCK)
       return DOSFALSE;
   pkt->bp_Pkt.dp_Arg1 = sect;
   if (SendPacket(pid,pkt) != DOSTRUE) return DOSFALSE;
   return WaitPacket(pkt);
}

/* ------------------------------------------------------------------
*/

/* Directory output formatting stuff.  Some of the operators are coded
   as macros to make them fast.  I try to keep the formatting stuff as
   consistent as possible between the two directory functions, so that
   a valid speed comparison between the two methods can be accurately
   drawn. */

#define NAMESIZ 23L                             /* Chars/Name */
#define PERLINE 3L                              /* Names/Text Line */
#define INDENT  4L                              /* Line indent */
#define LINESIZ (NAMESIZ*PERLINE+INDENT+1L)     /* Size of line buff */

char line[LINESIZ];                             /* The line buffer */

/* This macro returns TRUE if the line buffer is full. */

#define LineFull(p)     ((ULONG)((p)-line) >= LINESIZ-1L)

/* This macro returns TRUE if the line buffer if empty. */

#define LineEmpty(p)    ((p) <= line + INDENT)

/* This macro clears the line buffer. */

#define LineClear(p)    { \
   setmem(line, (unsigned)(LINESIZ-1L), ' '); \
   (p) = line + INDENT; }

/* This macro writes the buffer to the current output. */

#define LineWrite()     (Write(Stdout,line,LINESIZ))

/* ------------------------------------------------------------------
*/

/* This section contains the main functions. */

/* This function reads the given key, assumed to be the directory block,
   and builds the key table based on that directory block's hash table.  If
   the given block isn't a directory block, the function returns FALSE.
   Note that using the AddKey() function to add each entry is not all that
   efficient; it would be faster to fill the table and then run an
   efficient sort on that table.  This is left as an extension. */

BOOL ReadDir(pid,pkt,key)
struct MsgPort *pid;
BigPacket *pkt;
LONG key;
{
   short i;
   LONG *table;
   BLOCK *blk;

   if (ReadBlk(pid,pkt,key) != DOSTRUE) return FALSE;
   blk = (BLOCK *) pkt->bp_Data;
   if (blk->b_type != FT_SHORT_FILE) return FALSE;
   if (blk->b_sub != ST_ROOT_DIR
        && blk->b_sub != ST_USER_DIR)
                return FALSE;

   table = &(blk->b_hash[0]);
   keys[0] = 0L;
   for (i = 0; i < HASHSIZ; i++)
      if ((key = table[i]) != 0L)
          AddKey(key,0L);
   return TRUE;
}

/* This function implements the actual Fast Directory command.  It will be
   called after a little testing is done to insure that the given DOS
   device supports a normal trackdisk compatible file structure.  */
```

continued...

```
void FastDir(pid,pkt)
struct MsgPort *pid;
BigPacket *pkt;
{
   register BLOCK *blk;
   LONG key, collis;
   char *ptr;
   long len;

   blk = (BLOCK *) pkt->bp_Data;                /* Set up the block */
   *(line + LINESIZ-1L) = '\n';                 /* And the output buffer */
   LineClear(ptr);
   PopKey(key);
   while (ReadBlk(pid,pkt,key) == DOSTRUE) {
      len = (long) min(blk->b_name.len,NAMESIZ-1L);
      strncpy(ptr,&(blk->b_name.str[0]),len);
      *(ptr + len) = (blk->b_sub == ST_USER_DIR) ? '/' : ' ';
      ptr += NAMESIZ;
      if (LineFull(ptr)) {
         LineWrite();
         LineClear(ptr);
      }
      if ((collis = blk->b_collis) != 0L) {     /* Any collision? */
         if (CylEqu(collis,blk->b_key))
            key = collis;
         else {
            AddKey(collis,blk->b_key);
            PopKey(key);
         }
      } else if (top != 0L) {
         PopKey(key);
      } else
         break;
   }
   if (!LineEmpty(ptr)) LineWrite();
}

/* This function implements a normal directory command, also using packets,
   but the higher level EXAMINE_OBJECT and EXAMINE_NEXT packets, which
   should work for any DOS device capable of supporting files.  */

BOOL NormalDir(pid,lock)
struct MsgPort *pid;
struct FileLock *lock;
{
   register FIB *fib;
   BigPacket *pkt;
   BOOL once = FALSE;
   char *ptr;
   long len;

   *(line + LINESIZ-1L) = '\n';
   LineClear(ptr);
   pkt = AllocPacket(ACTION_EXAMINE_OBJECT,sizeof(FIB),FIBMEM);
   fib = (FIB *) pkt->bp_Data;
   pkt->bp_Pkt.dp_Arg1 = ((ULONG) lock) >> 2;
   /* Do First Examine, and check we've got a directory. */
   if (DoPacket(pid,pkt) != DOSTRUE || fib->fib_DirEntryType < 0L) {
      FreePacket(pkt,sizeof(FIB));
      return FALSE;
   }
   pkt->bp_Pkt.dp_Action = ACTION_EXAMINE_NEXT;
   while (DoPacket(pid,pkt) == DOSTRUE) {
      once = TRUE;
      len = (long) min(fib->fib_FileName[0],NAMESIZE-1L);
      strncpy(ptr,&(fib->fib_FileName[1]),len);
      *(ptr + len) = (fib->fib_DirEntryType > 0L) ? '/' : ' ';
      ptr += NAMESIZ;
      if (LineFull(ptr)) {
         Linewrite();
         LineClear(ptr);
      }
   }
   if (!once)
      Write(Stdout,NO_FILES,sizeof(NO_FILES));
   else if (!LineEmpty(ptr))
      LineWrite();
   FreePacket(pkt,sizeof(FIB));
   return TRUE;
}
/* ---------------------------------------------------------------------
*/
/* This is the main function.  It checks to make sure we're not called
from
   WorkBench first.  If not, then it gets a lock on the given directory
   name, if possible.  The lock yields the handler PID, which together
with
   an allocated block packet is sent to the ReadDir() function.  If the
   function is successful the validity of the file structure has been
   tested, and the key list can contain some keys.  FastDir() processes
the
   keys, if there are any.  If the ReadDir() failed, then there's
something
   funny about the file structure (RAM: is a good example of this), and
   thus NormalDir() is called to provide the desired directory listing.
*/

main(argc,argv)
int argc;
char *argv[];
{
   extern BPTR Output();                /* DOS "stdout" equivalent */
   struct FileLock *lock;               /* File Lock for directory */
   struct MsgPort *pid;                 /* PID from file lock */
   BigPacket *pkt;                      /* Packet for DOS Functions */
   char *dir;                           /* Directory name */
   BOOL norm = FALSE;                   /* Forced normal directory */
   if (argc == 0)                       /* Called from Workbench */
      Exit(RETURN_ERROR);
   if (argc >= 2 && argv[1][0] == '-') {
      dir = argv[2];
      norm = toupper(argv[1][1]) == 'N';
      trackonly = toupper(argv[1][1]) != 'C';
   } else
      dir = argv[1];
   Stdout = Output();
   if ((DosBase =
     (struct DosLibrary *)OpenLibrary("dos.library",33)) == NULL) {
      Write(Stdout,OLD_LIBRARY,sizeof(OLD_LIBRARY));
      Exit(1);
   }
   if ((lock =
     (struct FileLock *) BADDR(Lock(dir,SHARED_LOCK))) == NULL) {
      Write(Stdout,ERR_DIR,sizeof(ERR_DIR));
      Exit(RETURN_WARN);
   }
   pid = lock->fl_Task;
   pkt = AllocPacket(ACTION_GET_BLOCK,sizeof(BLOCK),BLKMEM);
   if (!norm && GetCyl(BADDR(lock->fl_Volume)) &&
       ReadDir(pid,pkt,lock->fl_Key)) {
      if (top != 0)
         FastDir(pid,pkt);
      else
         Write(Stdout,NO_FILES,sizeof(NO_FILES));
   } else if (!NormalDir(pid,lock))
      Write(Stdout,NOT_DIRECTORY,sizeof(NOT_DIRECTORY));
   FreePacket(pkt,sizeof(BLOCK));
   pkt = AllocPacket(ACTION_FREE_LOCK,0L,0L);           /* UnLock() */
   pkt->bp_Pkt.dp_Arg1 = ((ULONG)lock) >> 2;
   DoPacket(pid,pkt);
   CloseLibrary(DosBase);
   Exit(RETURN_OK);
}
```

•AC•

Roomers

As promised, the rumors are falling faster than the autumn leaves.

by the Bandito

Commodore's net income rose 75 percent during the fiscal fourth quarter, locking up a fifth consecutive quarterly profit. The net income was 2.1 million, roughly six cents a share. For the year, the profit was 89 cents a share, 28.6 million total. Word has it that the European market is making more money than the United States market.

Commodore is currently working on an advertising campaign for the Amiga 2000, touting it as a desktop publishing and desktop video machine. The Amiga 500 is in a serious back-ordered state, but Commodore hopes to move about 15,000 machines by the end of September, which is slightly short of the previous projection that they'll sell 70,000 by the end of the year. They hope to sell about 17,000 Amiga 2000 systems by then, too.

A big push for the Amiga in Germany, Norway and England is underway. Commodore is giving local dealers the once-over on the latest software and hardware.

Commodore is also assisting the development of the New York Institute of Technology's video digitizer. Insiders say the project is still months from completion.

Talk has also surfaced about a bug in AmigaDOS that hurt disk performance. Supposedly, a typo in the operating system code changed the disk caching algorithm from least-recently-used to most-recently-used, meaning disk performance isn't as good as it should be.

Rumors say CSA is working on a 68030-based Amiga for the government that increases performance even further than the CSA 68020 systems.

Word on the Magic Sac Macintosh emulator is that it is up and booting the Mac "sad face" screen, which means your Mac is sick and needs to see a dealer. Meanwhile, Data Pacific has added hard disk support to the Atari ST version, so such support may be added to the Amiga version, too.

The Commodore 64 emulator is still scheduled for release at the end of August, but company reps say they'll wait until every bug is crushed before they ship. List price is still $129.95. Turbo loads are not working at this point.

RJ Mical got word to the Bandito that the recent rumor about his game may be misleading. It may be true that Electronic Arts will not publish the game (once code-named "Ballgame"), but that doesn't mean it won't ever be published. He may find another publisher in the future. . .

Meanwhile, Mical and former Amiga Los Gatos hardware engineer Dave Needle have been hired by Epyx to work on new projects that are "non-software-based," thus leading the company in a new direction. Said RJ: "We're designing a solar-powered flashlight, but don't tell anyone, OK?" He is using the Amiga as a development station, however.

Recently, the Usenet Amiga group, comp.sys.amiga, was deluged with messages regarding Leo Schwab's VideoScape 3D animation that reproduced a scene from the "Red's Dream" movie at SIGGRAPH. Schwab, the author of display hacks such as "Robotroff" and "Viacom," brought the animation to the Aegis table in the Amiga booth. They played it on an Amiga 2000 with their music program, Sonix, doing the background music.

After the show, a representative of Pixar warned Schwab that he should not show the animation because the concept of a red, juggling unicycle is a copyright of Pixar, in much the same way that Disney restricts use of Mickey Mouse. Schwab worked carefully to avoid angering the Pixar reps, and all parties agreed to one last showing of Schwab's animation at a FAUG meeting. After it was all over, Schwab ended his conclusive posting with a definitive phrase: "Pixar is going to have competition at the SIGGRAPH Film and Video Show next year."

For some reason, Pixar also took a jab at the Amiga in a paper they presented at the show. They were discussing a ray-tracing program that was rendering an image of moving Jello. Pixar said they had a roomful of Amigas working on the problem. . .

The Amiga is featured in a new movie "Disorderlies," which features the rap group the Fat Boys. Some people thought they saw the Amiga in the background in a laboratory in the latest James Bond movie, "The Living Daylights."

continued on page 66

Amiga Artist: Brian Williams

by John Foust

If you are a fan of Amiga art, you certainly recognize the name Brian Williams.

Williams is a senior at Benedictine College in Lisle, Illinois, majoring in computer science. He has worked on artwork for past Commodore projects. Along with programmer Glenn Tenney (known for porting Electronic Arts games) and *Defender of the Crown* artist Jim Sachs, Williams created images for the Amiga 2000 dealer demo disk shown at Spring COMDEX. Sachs worked on the "attract mode" of the demo; the initial screens to lure unsuspecting buyers to the machine.

The demonstration tells the story of ancient Egyptians who use the Amiga to present the Pharoah with plans for a giant sandstone Amiga 'A.' The ancient planners use desktop publishing and CAD programs in their design. The demo shows off the Amiga's sound, graphics and multitasking. The demo is quite long, consuming the equivalent of two or three disks of data.

As a programmer, Williams had hoped to create computer-aided design and paint programs for the Amiga. . . instead, he turned to artwork. He started by making picture disks for his local Amiga dealer. Soon, Amiga owners requested copies of his pictures. Williams prepared several disks of images and distributed them freely.

When Williams first saw the Amiga, the Preferences program impressed him as much as a beta version of Deluxe Paint. "Preferences blew me away. It showed full video because you could move the screen beyond the regular video area. I felt confident that someday Deluxe Paint would paint out there."

Williams also thinks software developers have been slow to realize the needs of Amiga artists. In particular, he emphasizes the need for programs that draw in the border regions of the screen (often called the overscan area). "Now we have full video in Deluxe Paint. Why wasn't everything full video? Because no one asked for it."

Nonetheless, Williams is very happy with some existing tools. "Deluxe Paint is an incredible tool. It is better than 90 percent of what I've seen on other systems. It is almost as if it [D-Paint] is an extension of myself; it has such a fluid user interface. That [interface] is something that no other Amiga program has captured." Williams remains hopeful that other programs will go beyond Deluxe Paint. "There are no programs out there yet that express the full potential of this machine."

What techniques does Williams use to draw a picture? "First, you know what you are trying to draw. I don't usually touch the color palette; I optimize it later on. I start by blocking out light and shadow (to get the outlines), then fill-in and add shading." Williams then switches to magnify mode to dither and anti-alias lines. "Contrast—That's all-important. You can't think of 'an object here, an object here.' Light and shadow are very important."

"I think a lot of people don't like drawing on a computer. They find it difficult because they like to think they are drawing on paper. They try to draw a straight line and they expect to lift the pencil to make the line thinner; I expect jagged lines. My lines are already dithered in my head."

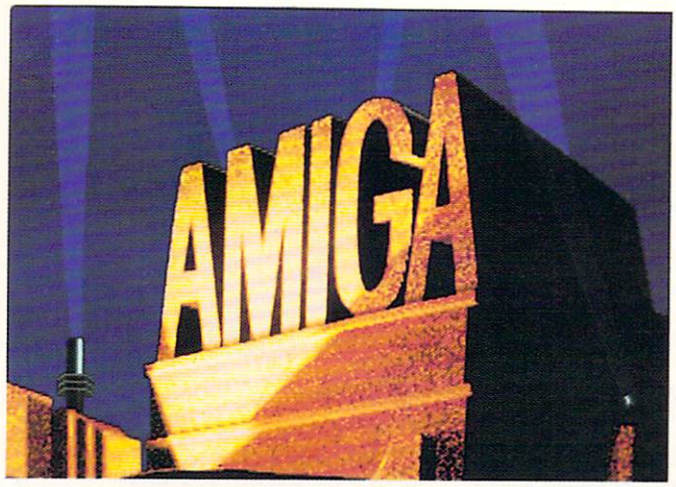
Williams spends most of his time in magnify mode, adjusting thousands of pixels, one by one, to get the desired effect. "I have enough inspiration to carry me through the long parts. Often, my drawings are 50 percent finished after an hour. The time afterwards makes all the difference. If you expect it to be easy, then it will be hard."

Some of Williams' pictures are so life-like, so detailed, that people wonder if they are digitized. He owns a Digi-View, but does not use it to produce his artwork. "Digitizers can be used or misused. They make it very easy for people to bring real world objects into their Amiga."

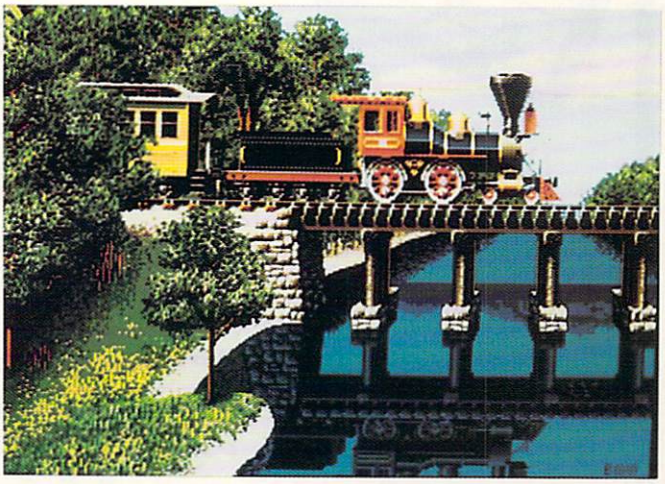
Does Williams consider digitized images to be artwork? "Sometimes. I ask the person, 'What have you done with it? Is it an original still-life that has some meaning or is it a picture from Popular Mechanics?' Nothing disappoints me more than downloading an image and finding that someone digitized something and touched it up. It requires a good eye to make a good picture."

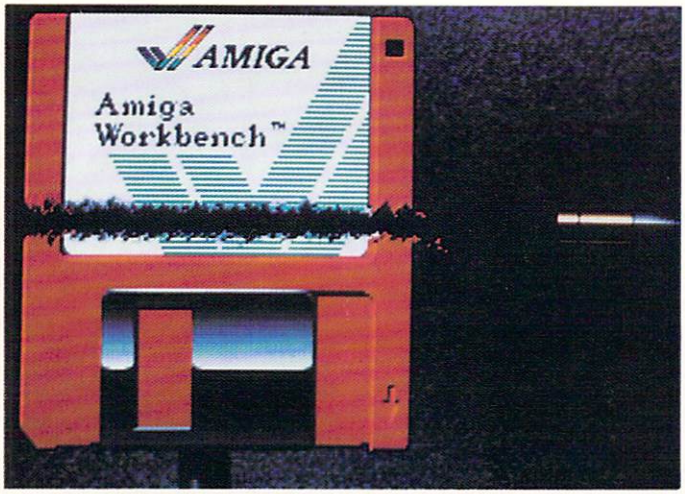
"On the Amiga, I see the digitizer as good for inputting very basic images. Once it's input, that is only the beginning — You start pulling it apart, you redefine the image. For me, the digitized image is only the start of a lot of hours of work."

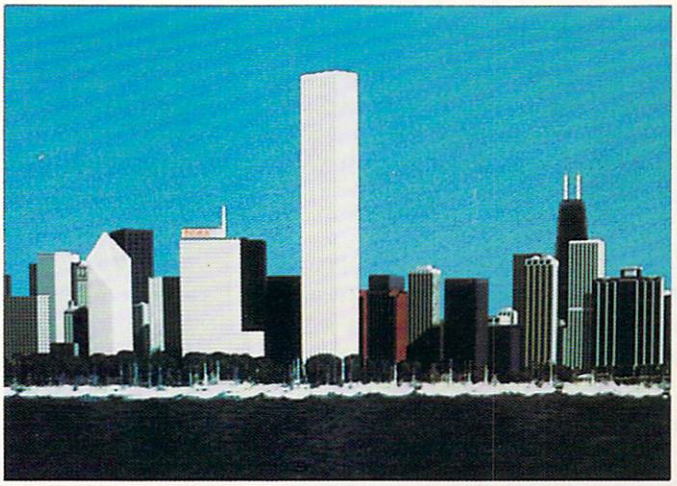
•AC•

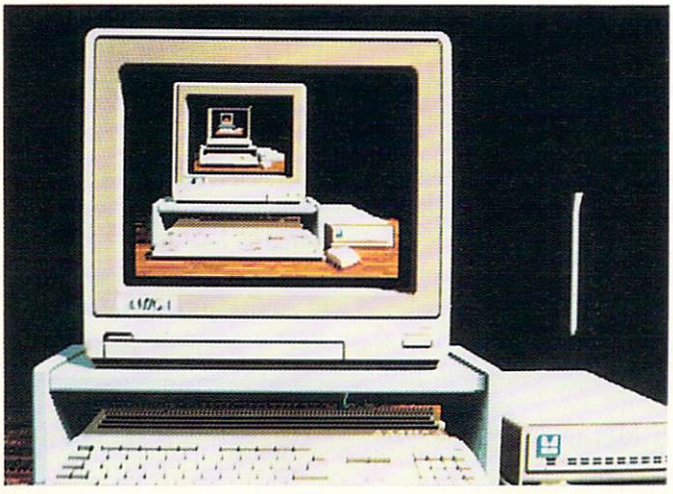


Showcased here are just five of the many fine works of art by Brian Williams.









(continued from page 63)

The Mytech software company has ported the COMAL language to the Amiga. Reportedly, it leaves about 150 K of memory for programs in a 512 K machine. No word on when it will be out.

Digital Creations, the Gizmos people, are working on several projects: a $700 professional quality genlock, a video digitizer, a HAM paint utility and a program that prints IFF pictures as wall-sized posters.

MicroIllusions is said to working on new programs, including an advanced animation program and a music program called Music-X.

"Rumormongers" venture that Broderbund is developing software for the Amiga. In their latest catalog, Broderbund used Amiga screen shots to show off sample screens from their Apple IIGS programs. They have also hired some Amiga-experienced programmers.

Aegis Animator author Jim Kent is working on an Atari ST program for Antic Software called CyberPaint. Antic may also be coming out with another animation program for the ST. Has Kent abandoned the Amiga? Not really, he did some of the compression routines for the yet-unreleased Amiga Live!.

Lattice says their latest C compiler, version 4.0, will be out in October. Jim Goodnow showed his Amiga source-level debugger at a recent BADGE meeting. He also said version 4.1 of the Manx C compiler will be shipping this fall.

A new Modula-2 compiler is available. From the accounts I've seen on the networks so far, it sounds great. The bugs are few and far between, which is a far cry from the TDI compiler. The manual is huge and several libraries of routines are included. A review should appear in this magazine very soon. The new compiler, priced at $199, is distributed by Oxxi (the people who market MaxiPlan) (714) 999-6710.

A company called the Gemstone Group has a low-cost 68020 / 68881 upgrade board for the Amiga. For more information, call (312) 537-7405.

At last word, news came that Atari has bought the Federated consumer electronics chain. Federated is one of Commodore's biggest sellers of Commodore 64, 128 and Amiga products. Will Atari let them sell Commodore products?

•AC•

Programming Modula-2

with the Amiga™

Fast File I/O

A module to call the DOSFiles Read and Write procedures *without going through Streams.*

by Steve Faiwiszewski

Probably the easiest method of doing I/O in Modula-2 is to use the InOut module. Unfortunately, in the TDI package, I/O done through InOut can be painfully slow. The reason for the lack of speed is that for every character to be read (or written), the InOut module calls the ReadChar procedure from module Streams, which in turns calls the AmigaDOS Read (or Write) procedure (declared in the DOSFiles module).

The extra level of indirection (going through the Streams module) and all the procedure calls required for the reading/writing of each byte add up to a great deal of overhead which affects performance tremendously. Well, there is a better way (why else would I be talking about it?).

The "better way" requires that you have a module to call the DOSFiles Read and Write procedures *without going through Streams*. In addition, instead of calling Read (or Write) for every character, the I/O should be buffered, meaning that chunks of bytes will be read or written, rather than just single characters. Far fewer calls to Read will be necessary, and the overhead will be reduced significantly.

The programs using this 'fast I/O' module need not know anything about buffering; they simply call the 'fast' read and write procedures. Such a module (called FastFileIO) appears in the following source listing. A few points of interest regarding FastFileIO follow.

Notice how the type 'FastFile' is declared in the definition module. Such a type is known as an opaque, since any other module cannot examine the component parts of FastFile. FastFile is fully declared in the implementation module. This practice of concealing the gory details is known as 'data hiding' and is encouraged in Modula-2. Any client module (i.e. any module that makes use of FastFileIO) will be able to pass only variables of type FastFile as parameters to procedures and to assign one FastFile variable to another.

Notice in the procedure 'FastRead' how bytes are copied directly from the FastFile's internal buffer to the spot in memory pointed to by the 'Buffer' parameter (the opposite is true for FastWrite). Instead of using arrays (i.e. copy from one array to another), I use pointers. Pointers allow for greater flexibility, while also avoiding the size constraints that would be introduced if arrays had been used. **A note of caution:** I take advantage of TDI's non-standard ability to use the INC procedure to increment pointers (and addresses). The original definition of Modula-2 specifies that INC can be used only on scalar types (such as INTEGER, CARDINAL, CHAR, subranges and enumerated types), so other Modula compilers might consider using INC on pointers to be an error.

Just how much faster is this FastFileIO module? The following listings include two simple file copiers — one using InOut and one using FastFileIO. Both programs read one byte at a time from the input file and write it to the output. The difference in speed is impressive — the 'SlowCopy' program takes more than 15 seconds to copy a 2K file; the 'BetterCopy' program takes less than a second to complete the same task. Note that the 'BetterCopy' can still be improved. Instead of reading one character at a time, it could read blocks of characters, thus decreasing overhead and improving overall performance.

Late Breaking News

By the time you read this article, a new Modula-2 compiler should be available for the Amiga. This package, called the Benchmark Modula-2 Software Construction Set, is distributed by Oxxi, Inc. (Tel: 714–999–6710). An in-depth review should appear in these pages shortly, so all I'll say now is that this one-pass compiler is lightning fast — when it comes to compile and link speed, the new compiler puts all other Amiga compilers to shame.

continued...

```
DEFINITION MODULE FastFileIO;

FROM SYSTEM IMPORT ADDRESS;

TYPE
    FastFile;

VAR
    FastDone : BOOLEAN;

(* Note:  All the following procedures modify the FastDone *)
(*       variable.  FastDone is TRUE if the operation was *)
(*       completely successful, and FALSE otherwise.      *)

PROCEDURE FastOpenInput(VAR F : FastFile;
                        VAR name : ARRAY OF CHAR;
                        Size : CARDINAL);
(* Open file for input.                                    *)
(* name : the name of the file to open for input. *)
(* Size : specifies the size of the buffer to be used   *)
(*       internally for reading.  The larger the buffer    *)
(*       the less the overhead involved in reading.        *)
(*           FastDone is set appropriately.              *)

PROCEDURE FastOpenOutput(VAR F : FastFile;
                         VAR name : ARRAY OF CHAR;
                         Size : CARDINAL);
(* Open file for output.                                   *)
(* name : the name of the file to open for output. *)
(*Size : specifies the size of the buffer to be used *)
(* internally for writing.  The larger the buffer      *)
(* the less the overhead involved in writing.          *)
(* FastDone is set appropriately.                      *)

PROCEDURE FastClose(VAR F : FastFile);

PROCEDURE FastReadChar(F : FastFile; VAR c : CHAR);
(* Read on character from file F and place it in c.   *)
(* File F must be already open for input.              *)
(* FastDone is set appropriately.                     *)

PROCEDURE FastReadLine(F : FastFile; VAR line : ARRAY OF
CHAR);
(* Read a string of characters from file F and place them
   in line.  The reading of characters is terminated when
   an End-Of-Line character is encountered, or line become
   full.
*)
(* File F must be already open for input.              *)
(* FastDone is set appropriately.                     *)

PROCEDURE FastRead(F : FastFile; Buffer : ADDRESS;
                   Length : LONGCARD): LONGCARD;
(* Read bytes and place them in memory starting   *)
(* at the address pointed to by Buffer.              *)
(* Length specifies how many bytes to read.         *)
(* Returns the actual number of bytes read.          *)
(* File F must be already open for input.             *)
(* FastDone is set appropriately.                     *)

PROCEDURE FastWriteChar(F : FastFile; VAR c : CHAR);
(* Write character c to output file F.         *)
(* File F must be already open for output.   *)
(* FastDone is set appropriately.               *)

PROCEDURE FastWrite(F : FastFile; Buffer : ADDRESS;
                    Length : LONGCARD): LONGCARD;
(* Write bytes from memory, starting at the address *)
(* pointed to by Buffer, to file F.              *)
(* Length specifies how many bytes to write.          *)
(* Returns the actual number of bytes written.        *)
(* File F must be already open for output.             *)
(* FastDone is set appropriately.                     *)

PROCEDURE FastEOF(F : FastFile): BOOLEAN;
(* Returns true of end of file was encountered *)
(* for input file F.  This procedure has no      *)
(* meaning for output files.                          *)

END FastFileIO.


MPLEMENTATION MODULE FastFileIO;

(*$T-*) (* Disable Subscript checking.  Not needed here *)
(*$Q+*)

FROM SYSTEM       IMPORT NULL, ADR, ADDRESS, TSIZE;
FROM Memory       IMPORT AllocMem, FreeMem, MemPublic,
MemReqSet;
FROM DOSFiles IMPORT Open, Close, Write, Read, ModeOldFile,
                     ModeNewFile, FileHandle;
FROM DOSLibrary  IMPORT DOSName, DOSBase;
FROM Libraries   IMPORT OpenLibrary;


CONST
     EOL = 12C; (* LF ends a line *)

TYPE
     CharPtr = POINTER TO CHAR;

     FastFile = POINTER TO FastFileRec;

     FastFileRec = RECORD
                      IndexPtr,
                       bufp    : CharPtr;
                      BufSize: CARDINAL;
                      mode,
                       length : LONGINT;
                        fh      : FileHandle;
                        eof     : BOOLEAN;
                   END;
```

```
PROCEDURE Flush(F : FastFile): BOOLEAN;
(* Write out the file's buffer to disk *)
VAR
    size,
    ret : LONGINT;
BEGIN
    WITH F^ DO
        size := LONGINT(ADDRESS(IndexPtr) - ADDRESS(bufp));
        ret := Write(fh,bufp,size);
        IF (ret <= 0) OR (size <> ret) THEN RETURN FALSE
END;
        length := 0;
        IndexPtr := bufp
    END;
    RETURN TRUE
END Flush;

PROCEDURE FillBuffer(F : FastFile);
(* fill the file's buffer from bytes read from disk *)
VAR
    i : CARDINAL;
BEGIN
    WITH F^ DO
       length := Read(fh,bufp,LONGINT(BufSize));
       IF length <= 0 THEN
           eof := TRUE
      ELSE
           IndexPtr := bufp
       END;
    END;
END FillBuffer;

PROCEDURE CommonOpen(VAR F : FastFile;
                     VAR name : ARRAY OF CHAR;
                     Size : CARDINAL; Mode : LONGINT):
BOOLEAN;
VAR
    tmp : FileHandle;
BEGIN
    tmp := Open(name,Mode);
    IF tmp = 0 THEN RETURN FALSE END;
    F := AllocMem(TSIZE(FastFileRec),MemReqSet{MemPublic});
    IF F = NULL THEN
    (* Oh oh! Not enough memory for another FastFileRec *)
      Close(tmp);
      RETURN FALSE;
    END;
    WITH F^ DO
        bufp := AllocMem(LONGCARD(Size),MemReqSet{MemPublic});
        IF bufp = NULL THEN
      (* Not enough memory for our buffer. Better cleanup *)
          Close(tmp);
          FreeMem(F,TSIZE(FastFileRec));
          RETURN FALSE
       END;
       IndexPtr := bufp;
       length := 0;
       fh := tmp;
       eof := FALSE;
       BufSize := Size;
       mode := Mode;
    END;
    RETURN TRUE
END CommonOpen;

PROCEDURE FastOpenInput(VAR F : FastFile;
                        VAR name : ARRAY OF CHAR;
                        Size : CARDINAL);
BEGIN
    IF CommonOpen(F,name,Size,ModeOldFile) THEN
       FillBuffer(F);
       FastDone := TRUE
    ELSE
       FastDone := FALSE
    END;
END FastOpenInput;
```

continued...

```
PROCEDURE FastOpenOutput(VAR F : FastFile;
                         VAR name : ARRAY OF CHAR;
                         Size : CARDINAL);
BEGIN
    FastDone := CommonOpen(F,name,Size,ModeNewFile)
END FastOpenOutput;

PROCEDURE FastClose(VAR F : FastFile);
(* Close the actual disk file, and release *)
(* all the memory allocated to it.          *)
BEGIN
    WITH F^ DO
        IF mode = ModeNewFile THEN
            FastDone := Flush(F)
        ELSE
            FastDone := TRUE
        END;
        Close(fh);
        FreeMem(bufp,LONGCARD(BufSize));
    END;
    FreeMem(F,TSIZE(FastFileRec));
END FastClose;

PROCEDURE FastReadChar(F : FastFile; VAR c : CHAR);
BEGIN
    WITH F^ DO
        IF mode = ModeOldFile THEN
            IF length <= LONGINT(ADDRESS(IndexPtr) -
ADDRESS(bufp)) THEN
                IF eof THEN
                    FastDone := FALSE;
                    RETURN
                END;
                FillBuffer(F);
                IF eof THEN
                    FastDone := FALSE;
                    RETURN
                END;
            END;
            c := IndexPtr^;
            INC(IndexPtr);
            FastDone := TRUE
        ELSE
            FastDone := FALSE
        END; (* if *)
    END; (* with *)
END FastReadChar;

PROCEDURE FastWriteChar(F : FastFile; VAR c : CHAR);
VAR
    stat : BOOLEAN;
BEGIN
    WITH F^ DO
        IF mode = ModeNewFile THEN
            FastDone := TRUE;
            IF CARDINAL(ADDRESS(IndexPtr) - ADDRESS(bufp)) =
BufSize THEN
                FastDone := Flush(F)
            END;
            IndexPtr^ := c;
            INC(IndexPtr);
        ELSE
            FastDone := FALSE
        END
    END;
END FastWriteChar;

PROCEDURE FastRead(F : FastFile; Buffer : ADDRESS;
                                 Length : LONGCARD): LONGCARD;
VAR
    TmpBuf : CharPtr;
    TmpIndex : CARDINAL;
    TmpLength,
    i,
    CharsToMove,
    CharsInBuffer : LONGCARD;
BEGIN
    TmpLength := Length;
    TmpBuf := CharPtr(Buffer);
    TmpIndex := 0;
    WITH F^ DO
        IF mode = ModeOldFile THEN
            WHILE (TmpLength > 0) AND NOT eof DO
                CharsInBuffer := LONGCARD(length) -
                                 LONGCARD(ADDRESS(IndexPtr) -
                                 ADDRESS(bufp));
                IF TmpLength > CharsInBuffer THEN
                    CharsToMove := CharsInBuffer
                ELSE
                    CharsToMove := TmpLength
                END;
                FOR i := 1 TO CharsToMove DO
                    TmpBuf^ := IndexPtr^;
                    INC(IndexPtr);
                    INC(TmpBuf)
                END;
                DEC(TmpLength,CharsToMove);
                IF length = LONGINT(ADDRESS(IndexPtr) -
                            ADDRESS(bufp)) THEN
                    FillBuffer(F)
                END;
            END;
            DEC(Length,TmpLength);
            FastDone := (TmpLength = 0);
            RETURN Length
        ELSE
            FastDone := FALSE;
            RETURN 0
        END
    END;
END FastRead;
```

```
PROCEDURE FastWrite(F : FastFile; Buffer : ADDRESS;
                          Length : LONGCARD): LONGCARD;
VAR
    i,
    RoomInBuffer,
    CharsToMove,
     TmpLength : LONGCARD;
     TmpIndex  : CARDINAL;
     TmpBuf    : CharPtr;
     stat      : BOOLEAN;
BEGIN
    TmpLength := Length;
    TmpBuf := CharPtr(Buffer);
    TmpIndex := 0;
    stat := TRUE;
    WITH F^ DO
        IF mode = ModeNewFile THEN
            WHILE (TmpLength > 0) AND stat DO
               RoomInBuffer := LONGCARD(BufSize) -
                               LONGCARD(ADDRESS(IndexPtr) -
                               ADDRESS(bufp));
                IF TmpLength > RoomInBuffer THEN
                  CharsToMove := RoomInBuffer
                ELSE
                   CharsToMove := TmpLength
                END;
                  FOR i := 1 TO CharsToMove DO
                     IndexPtr^ := TmpBuf^;
                    INC(IndexPtr);
                    INC(TmpBuf)
                END;
                DEC(TmpLength,CharsToMove);
                 IF CARDINAL(ADDRESS(IndexPtr) - ADDRESS(bufp))
=
                   BufSize THEN
                     stat := Flush(F)
                END;
            END;
            DEC(Length,TmpLength);
              FastDone := stat AND (TmpLength = 0);
            RETURN Length
        ELSE
            FastDone := FALSE;
            RETURN 0
        END
    END;
END FastWrite;

PROCEDURE FastEOF(F : FastFile): BOOLEAN;
BEGIN
    RETURN F^.eof
END FastEOF;

PROCEDURE FastReadLine(F : FastFile; VAR line : ARRAY OF
CHAR);
VAR
    i : CARDINAL;
BEGIN
    i := 0;
    WITH F^ DO
        IF mode = ModeOldFile THEN
           REPEAT
                IF NOT eof THEN
                     line[i] := IndexPtr^;
                    INC(IndexPtr);
                    INC(i);
                  IF length = LONGINT(ADDRESS(IndexPtr) -
     ADDRESS(bufp)) THEN
                        FillBuffer(F)
                    END;
                END;
        UNTIL eof OR (line[i-1] = EOL) OR (i = HIGH(line));
            IF line[i-1] = EOL THEN
                line[i-1] := 0C
            END;
             line[i] := 0C;
              FastDone := NOT eof;
        ELSE
            FastDone := FALSE
        END;
    END; (* with *)
END FastReadLine;

BEGIN
   (* Make sure that DOS library is opened *)
    IF DOSBase = NULL THEN
       DOSBase := OpenLibrary(DOSName,0);
    END;
END FastFileIO.


MODULE SlowCopy;

(*******************************************************)
(* This is a small, slow (and stupid) copy program,*)
(* which goes through InOut to do its I/O.         *)
(* Feel free to use this code as you please.   You *)
(* don't even have to keep my name in it.   Just don't *)
(* blame me for it!                                    *)
(*******************************************************)

FROM InOut          IMPORT OpenInput, OpenOutput, CloseInput,
                           CloseOutput, Read, Write, Done,
                           WriteString, WriteLn;
FROM DOSLibrary   IMPORT SIGBreakC;
FROM Tasks          IMPORT TaskPtr, FindTask;
IMPORT Trapper;

VAR
    c         : CHAR;
    Myself    : TaskPtr;
    Ldone     : BOOLEAN;
```

continued...

```
PROCEDURE CtrlC(): BOOLEAN;
(* see if control-c signal has arrived *)
BEGIN
    RETURN SIGBreakC IN Myself^.tcSigRecvd
END CtrlC;

BEGIN
    Myself := FindTask(0);
     WriteString('                              Slow File Copier');
    WriteLn; WriteLn;
     OpenInput(''); (* prompt for, and open input file *)
     OpenOutput(''); (* prompt for, and open output file *)
    Ldone := Done;
    WHILE Ldone AND NOT CtrlC() DO
       Read(c);
         Ldone := Done; (* we have to save the value of    *)
         Write(c)         (* Done because Write changes it. *)
    END;
    CloseInput;
    CloseOutput;
END SlowCopy.


MODULE BetterCopy;

    (*****************************************)
     (* Here's a slightly improved copy program. *)
     (* Instead of going through InOut, we read  *)
     (* characters using the FastFileIO module.  *)
     (* Speed can still be improved by changing  *)
     (* the code to read blocks of characters,   *)
     (* instead of one character at a time     *)
    (*****************************************)

FROM FastFileIO   IMPORT FastOpenInput, FastOpenOutput,
                         FastClose, FastReadChar,
                         FastWriteChar, FastEOF, FastFile,
                         FastDone;
FROM InOut          IMPORT WriteLn, WriteString, ReadString;
FROM DOSLibrary   IMPORT SIGBreakC;
FROM Tasks          IMPORT TaskPtr, FindTask;
IMPORT Trapper;

CONST
    EOL = 12C;
    BlockBufferSize = 544;
     CACHESIZE = BlockBufferSize * 2; (* holds 2 block buffs
*)

TYPE
    ModeType = (input,output);
VAR
    FastOutput,
     FastInput : FastFile;
     c : CHAR;
     dummy : BOOLEAN;
     Myself  : TaskPtr;

PROCEDURE CtrlC(): BOOLEAN;
(* see if control-c signal has arrived *)
BEGIN
     RETURN SIGBreakC IN Myself^.tcSigRecvd
END CtrlC;

PROCEDURE OpenFile(VAR F: FastFile; mode: ModeType): BOOLEAN;
VAR
     name : ARRAY[0..64] OF CHAR;
     done : BOOLEAN;
BEGIN
     done := FALSE;
    REPEAT
         WriteString('Enter ');
          IF mode = input THEN
             WriteString('Input')
         ELSE
             WriteString('Output')
         END;
          WriteString(' File Name: ');
         ReadString(name);
          IF mode = input THEN
             FastOpenInput(F,name,CACHESIZE)
         ELSE
             FastOpenOutput(F,name,CACHESIZE)
         END;
     UNTIL FastDone OR CtrlC();
     RETURN FastDone;
END OpenFile;

BEGIN
     Myself := FindTask(0);
      WriteString('                                  Better File
Copier');
     WriteLn; WriteLn;
     IF OpenFile(FastInput,input) THEN
          IF OpenFile(FastOutput,output) THEN
               WHILE NOT FastEOF(FastInput) AND FastDone DO
                  FastReadChar(FastInput,c);
                  FastWriteChar(FastOutput,c)
               END;
               FastClose(FastOutput)
          END;
          FastClose(FastInput)
     END
END BetterCopy.
```

•AC•

Window I/O

C Routines for input from and output to an arbitrary Window on an arbitrary Screen

by Read Predmore

When running a C program from a CLI window, all routines (such as printf(), putc(), getc()) use the CLI for input and output. This method is satisfactory for C programs when you know the program will be invoked from the CLI, and not by double clicking on an icon. If separate windows are desired for input and output, a Console can be opened on the WorkBench screen.

In general, you must open a window on an arbitrary screen and attach a console to this window. The details of opening this console are discussed in the Rom Kernal Manual chapter on the Console Device. One of the ROM demonstration programs, 'cons.c', is available on Fred Fish Disk 5. The 'cons.c' program works fine, but has many details that I have incorporated into general purpose subroutines. These routines allow a console device to be opened with any TextFont and then allow input and output to the device. Finally, close the console in a straight-forward way, without having to mess with the details.

Table 1 summarizes the various input and output options available to the Amiga C programmer. Three modes of I/O are presented. The first mode is for printing to and getting input from the CLI where the program was started. This method cannot be used if the program is started from WorkBench with an icon, because a 'printf()' or 'puts()' call causes a software error and a visit from Mr. Guru. You might hope that the output would just be lost, but that is not what happens with WorkBench 1.1.

The second method, which can be used with a program started from the WorkBench, is used to open a CON: device and do the I/O through that window. As is summarized in Table 1, a pointer to a FILE, pFILE, is set equal to the result of an fopen() of a CON:. Inputting a character is then accomplished with getc(pFILE). Inputting a string is accomplished with fgets(buf,num,pFILE), where buf is a pointer to char, which was probably defined as:

```
char buf(80);
```

The maximum number of input characters is set by num. Formatted input can be done with the fscanf (pFILE,format,pointers) function. Note that the FILE pointer, pFILE, is the first parameter in the fscanf() routine and is the last parameter in the fgets(), putc() and fputs() routines. This inconsistency in C will never be changed, since these functions are defined in this manner in the Kerninghan and Ritchie bible, The C Programming Language.

Output is accomplished with the functions putc(ch,pFILE), fputs(buf,pFILE) and fprintf(pFILE,format,variables) for character, string and formatted output, respectively.

When you are finished using a WorkBench Console, it is closed with:

```
fclose(pFILE);
```

I have developed a set of analogous routines for input from and output to an arbitrary Window on an arbitrary Screen. The Window console is opened with:

```
console = AttachConsole(window, name);
```

In this case, console is a pointer to a Console structure, window is a pointer to a Window, and name is a pointer to a string. The Console structure is not an Amiga structure, but rather one I have defined that incorporates the input and output message ports that are required for console I/O.

The Console structure is defined in my header file WINDOW.H as:

```
/* Group all ports, message pointers     */
/* and buffers in a 'Console' structure. */

struct Console {
   struct  MsgPort  *WritePort;  /*  Defined in <exec/ports.h>.  */
   struct  IOStdReq *WriteMsg;   /*  Defined in <exec/io.h>.     */
   struct  MsgPort  *ReadPort;
   struct  IOStdReq *ReadMsg;
   char readbuffer(48);
};
```

continued...

Don't worry if you don't know about about Message Ports. I wrote these routines so that you wouldn't have to bother with that level of detail. If you are really curious, you are welcome to delve into these routines, but it is not required.

Input from a Window console is done with congetc(console) and congets(buf,num,console) for character and string input, respectively. Formatted input is done in two steps. First, congets(buf,num,console) is used to input a string, then sscanf(buf,format,pointers) decodes the input string and stores the results using the pointers to variables.

Output to a Window console is accomplished with conputc(ch,console) and conputs(buf,console) for character and string output. As for input, formatted output is done in two steps. First, sprintf(buf,format,variable) is used to make a formatted string in the string buffer, buf. This string is then output with conputs(buf,console).

These Console routines are illustrated in the WINDOW_IO.C demonstration program. The program I have developed opens a 640 by 200 screen with 3 bits planes for 8 colors. Three resizable windows are then opened with the Ruby-8, Sapphire-14 and Garnet-9 fonts. Three consoles are then opened in those windows.

A simple I/O demo is then done with the input to Console-0 being output to both Console-1 and Console-2. The first demonstration does character I/O using congetc() and conputc(). The Console-0 window must be active for this test. The character is echoed back to Console-0 by the congetc() function. The main program also gives the hexadecimal code for the input character and sends the character to both Console-1 and Console-2. A CTRL/S causes the program to move on to the next step and CTRL/C exits the program.

The second step uses string I/O with the congets() and conputs() functions. An entire line of characters can be input until a carriage return is encountered. This string is sent to Console-1 and Console-2. With the chosen Fonts (which have proportional spacing), the conputs() output will be much more compact than the same characters sent out with the conputc() function.

This proportional spacing usually looks nicer, but has a limitation. If, for example, a large font is used and only 55 characters fit on a line when conputc() is used, the conputs() function can only put 55 characters on a line. This limit holds true, even if these 55 characters only take up 2/3 of a line when proportionally spaced. This "feature" appears to be specific to the Amiga console I/O, not my implementation of it.

Also, if two strings of 20 characters are output with conputs(), without a newline '\n' at the end of the first string, a blank gap shows up between the two strings of characters. This gap is present because the starting position of the second string is calculated as if the first string was output character by character.

The index() function, which is part of the Manx C library, is used to test for a CTRL/S or CTRL/C in the input string. A CTRL/S skips to the next step and a CTRL/C exits. The next three steps use string input and a sscanf() on the input string to input integers, hexadecimals and floating point numbers. Each time, a CTRL-S is used to skip to the next step.

Window.h

The window.h header file starts with a number of include statements for the Amiga 'include' directory. The Console structure is then defined to include all ports, message pointers and an input buffer. The size of this structure is 64 bytes.

Next, a set of bit flags are defined, so that various libraries, screens, windows and window consoles can be opened with the initialize() routine. The flags are accumulated in the variable, cur_resource. This variable is then tested in the closedown() routine, so that all the system resources are released before the program exits.

Console routines

The following are detailed descriptions of each of my console routines:

AttachConsole()

The Console opening routine is AttachConsole(window, name), which involves many details. First, memory is allocated for a Console structure. This structure, struct Console, is defined in the 'window.h' file and currently has a size of 64 bytes. A zero is returned by AttachConsole() if memory cannot be allocated, or if one of the subsequent message ports cannot be opened.

Next, the string ".write" is concatenated to the console name and this string is used to create a WritePort. The address which is returned by the CreatePort() function is stored in the console->WritePort pointer. Next, this port address is used in a CreateStdIO() call and the returned pointer is stored in console->WriteMsg. The same thing is now done for the ReadPort and ReadMsg pointers.

So far, we have opened up two ports, but they have not been related to the desired window. This reaction is done by equating the WriteMsg io_Data pointer to the window pointer, and the io_Length to the size of the Window structure:

```
console->WriteMsg->io_Data   = (APTR) window;
console->WriteMsg->io_Length = sizeof(*window);
```

This initialized IO Standard Request structure, WriteMsg, is used to open a console.device with:

```
OpenDevice("console.device", 0, console->WriteMsg, 0);
```

If this opening is successful, the io_Device and io_Unit for the ReadMsg are equated to those devices for the WriteMsg.

Finally, a SendIO() command is done to put the first input character from the keyboard into console->readbuffer[0]. The address of the memory allocated for the Console structure is returned. Much work has been done just in preparation to get the first character from the keyboard, but subsequent I/O will be done in a straightforward manner, using the same format as other C I/O functions.

DetachConsole()

The closing of a Window Console involves closing the various resources which were opened with AttachConsole() and freeing up the Console structure memory.

conputc()

The character output routine conputc() is called with a character argument and a pointer to a Console structure. The

continued...

Summary of I/O possibilities

Via the CLI:

OPEN	Automatic opening of stdin, stdout and stderr
INPUT	
Character	getchar();
String	gets(buf);
Formatted	scanf(format,pointers);
OUTPUT	
Character	putchar(ch);
String	puts(buf);
Formatted	printf(format,variables);
CLOSE	Automatic closing of stdin, stdout and stderr

Via a WorkBench console:

OPEN	pFILE = fopen(constr,"w+");
INPUT	
Character	getc(pFILE);
String	fgets(buf,num,pFILE);
Formatted	fscanf(pFILE,format,pointers) ;
OUTPUT	
Character	putc(ch,pFILE);
String	fputs(buf,pFILE);
Formatted	fprintf(pFILE,format,variables);
CLOSE	fclose(pFILE);

Via a window console:

OPEN	console = AttachConsole(window,name);
INPUT	
Character	congetc(console);
String	congets(buf,num,console);
Formatted	congets(buf,num,console); sscanf(buf,format,pointers);
OUTPUT	
Character	conputc(ch,console);
String	conputs(buf,console);
Formatted	sprintf(buf,format,variables); conputs(buf,console);
CLOSE	DetachConsole(console);

Where:

```
char   *buf, ch;
struct Console *console;
FILE   *pFILE;
int    num;
```

constr is a string such as "CON:0/10/640/110/console name"
format is a string such as "i=%d, x=%f, string=%s, char=%c \n"

Pointers are pointers to variables
or the addresses of variables, such as &i.

WriteMsg pointer in the Console structure is used to initialize io_Command to CMD_WRITE, io_Data to the address of the character, and io_Length to 1 (since only one character is being written). Then, DoIO() is called to actually output this character.

conputs()

The string output routine conputs() works in the same manner as conputc(), except that a pointer to a null-terminated string and a pointer to a Console structure are the two function parameters. The elements of the WriteMsg structure are initialized for a write command, io_Data is set to the string pointer, and an io_Length of -1 is used to signify that a null-terminated string is being sent. DoIO() is then called.

congetc()

The only argument for congetc() is a pointer to the Console structure which the input is coming from. It waits until a message is available on the console->ReadMsg and then stores the input character in a temporary location, while the ReadMsg channel is readied for getting the next character with CMD_READ and SendIO().

References

B.W. Kernighan and D.M. Richie, THE C PROGRAMMING LANGUAGE, Prentice -Hall, Inc., Englewood Cliffs, New Jersey 07632, 1978.

Manx AZTEC C68K, Version 3.20a for the AMIGA, Section on SYSTEM INDEPENDENT FUNCTIONS.

ROM Kernal Manual, Chapter on the Console Device, Commodore-Amiga, 1986. Waite, M., S. Prata and D. Martin, C PRIMER PLUS, H.W. Sams & Co. Indianapolis, Indiana, 1984.

Listing One
Window I/O Makefile

```
CFLAGS= -s +l +iram:defs

ALL=window_io.o

all:    foo $(ALL)
        del ram:defs
        copy df0:LIB/m32.lib ram:
        copy df0:LIB/c32.lib ram:
        ln -W $(ALL) ram:m32.lib ram:c32.lib
        del  ram:m32.lib ram:c32.lib
        @echo "All done!!"

foo:    defs
        copy defs ram:defs

defs:
        cc +hdefs -a +l window.h
```

Listing Two
Window.h

```
/* WINDOW.H
 *
 *   Read Predmore
 *   COPYRIGHT 1987
 *
 */

#include "intuition/intuition.h"

#include "exec/exec.h"
#include "exec/io.h"
#include "exec/memory.h"
#include "exec/types.h"

#include "graphics/gfxmacros.h"
#include "graphics/copper.h"
#include "graphics/gels.h"
#include "graphics/regions.h"
#include "graphics/clip.h"

#include "hardware/dmabits.h"
#include "hardware/custom.h"
#include "hardware/blit.h"

#include "devices/console.h"
#include "devices/keymap.h"

#include "libraries/dos.h"
#include "libraries/diskfont.h"
```

```
#include "libraries/mathffp.h"

#include "stdio.h"

/*   Group all ports, message pointers
   and buffers in a 'Console' structure. */

struct Console
{     struct   MsgPort              *WritePort;
          struct   IOStdReq *WriteMsg;
          struct   MsgPort              *ReadPort;
          struct   IOStdReq *ReadMsg;
          char     readbuffer[48];
};
/*-----------*/
/* MASK definitions */

#define  F_INTUITION  0x000001
#define  F_GRAPHICS   0x000002
#define  F_MATH         0x000004
#define  F_MATHTRANS  0x000008
#define  F_DOS          0x000010
#define  F_DISKFONT   0x000020

#define  F_SCREEN      0x000100
#define  F_MENUSTRIP  0x000200
#define  F_RASTER      0x000400

#define  F_WINDOW0     0x001000
#define  F_WINDOW1     0x002000
#define  F_WINDOW2     0x004000
#define  F_WINDOW3     0x008000

#define  F_CONSOLE0    0x010000
#define  F_CONSOLE1    0x020000
#define  F_CONSOLE2    0x040000
#define  F_CONSOLE3    0x080000

#define  F_FONT0        0x100000
#define  F_FONT1        0x200000
#define  F_FONT2        0x400000
#define  F_FONT3        0x800000

/*   Defines for attaching a console.   */
#define  CON_WPORT     0x01
#define  CON_WMSG      0x02
#define  CON_RPORT     0x04
#define  CON_RMSG      0x08
#define  CON_DEVICE    0x10


/* SCREEN AND WINDOW DEFINITIONS */

#define  DEPTH            3
#define  SHEIGHT       200     /*   SCREEN HEIGHT */
#define  SWIDTH        640

/*   Flags for function keys.   */
#define  CONPUTC_ON    0x0001L
#define  CONPUTS_ON    0x0002L
#define  KEYMAP_ON     0x0010L
#define  RAWIN_ON      0x0020L
#define  TEXT_TEST     0x0100L

/*   Keycodes from RKM section on the console device.
 *   Bit 8 is set high which corresponds to
 *   releasing the indicated key.
 *   Bit 9 has been added and set high as
 *   is done in the DECODE_CSI()
 *   routine.
 */
#define  F1_KEY         0x01D0L
#define  F2_KEY         0x01D1L
#define  F3_KEY         0x01D2L
#define  F4_KEY         0x01D3L
#define  F5_KEY         0x01D4L
#define  F6_KEY         0x01D5L
#define  F7_KEY         0x01D6L
#define  F8_KEY         0x01D7L
#define  F9_KEY         0x01D8L
#define  F10_KEY        0x01D9L
#define  HELP_KEY       0x01DFL
#define  UP_ARROW       0x01CCL
#define  DOWN_ARROW     0x01CDL
#define  RIGHT_ARROW    0x01CEL
#define  LEFT_ARROW     0x01CFL
```


Listing Three
Window_IO.c

```
/* WINDOW_IO.C
 *   COPYRIGHT   1986, 1987
 *   By Read Predmore
 *   window_io.c - a console device which can be
 *   attached to any window.
 */

#include "window.h"

#define   CTRL_C   0x03
#define   LF         0x0a
#define   CR         0x0d
#define   CTRL_S   0x13
ULONG   cur_resource = 0L;
ULONG DosBase;
ULONG DiskfontBase;
ULONG IntuitionBase;
ULONG GfxBase;
struct TextFont   *tfont[3];
struct   TextAttr tattr[3] = {
     { (unsigned char *)"ruby.font",       8,0,0 },
     { (unsigned char *)"sapphire.font",14,0,0 },
     { (unsigned char *)"garnet.font",    9,0,0 }
};
```

continued...

```
/* SCREEN AND WINDOW STRUCTURES */

struct NewScreen screen0 =
{
     0 ,                                /* LeftEdge          */
     0 ,                                /* TopEdge           */
     SWIDTH,                           /* Width             */
     SHEIGHT,                          /* Height            */
     DEPTH ,                            /* Depth             */
     0 ,                                /* DetailPen         */
     1 ,                                /* BlockPen          */
     HIRES,                            /* ViewModes         */
     CUSTOMSCREEN ,                  /* Type              */
     NULL ,                             /* *Font             */
    (UBYTE *)
   "WINDOW I/O TEST by Read Predmore - COPYRIGHT 1986, 1987",
     NULL ,                            /* *Gadgets          */
     NULL                               /* *CustomBitMap     */
};

struct NewWindow nw[3] =
{
    {
         0, 10,              /* starting position (left,top)  */
         640, 55,           /* width, height */
         1,2,                /* detailpen, blockpen */
         0,                   /* flags for idcmp */
        /* window gadget flags */
        WINDOWDEPTH | WINDOWDRAG
      | WINDOWSIZING | SMART_REFRESH | ACTIVATE,
         0,                   /* pointer to 1st user gadget */
         NULL,               /* pointer to user check */
        (UBYTE *)"Console-0",/* title */
         NULL,               /* pointer to window screen */
         NULL,               /* pointer to super bitmap */
         160,50,            /* min width, height */
         320,200,           /* max width, height */
        CUSTOMSCREEN
    },
    {
         0,65, 640, 68,   7,3,    0,
        WINDOWDEPTH | WINDOWDRAG |
        WINDOWSIZING | SMART_REFRESH,
         0,   NULL,   (UBYTE *)"Console-1",   NULL,   NULL,
        160,50,    320,200,
       CUSTOMSCREEN
    },
    {
         0,133,    640, 67,    5,4,    0,
        WINDOWDEPTH | WINDOWDRAG |
        WINDOWSIZING | SMART_REFRESH,
         0,   NULL,   (UBYTE *)"Console-2",   NULL,   NULL,
        160,50,    320,200,
       CUSTOMSCREEN
    },
};

struct Screen     *ps0;
struct Window     *pw[3];
struct RastPort  *rp[3];
struct Console   *console[3];

extern void          *OpenLibrary();
extern struct Console         *AttachConsole();
extern struct IntuiMessage  *GetMsg();
extern void                        *CreateStdIO();
extern void                        *CreatePort();
extern struct Screen        *OpenScreen();
extern struct TextFont        *OpenDiskFont();
extern struct Window           *OpenWindow();

long Func_flags = 0,
Ctrl_flags = 0;
char CSI_buf[80], In_buf[80];
int Print_OK = FALSE, CSI_flag = FALSE;


/* MAIN */
main(argc, argv)
int argc;
char *argv[];
{
     static char buffer[100], buffer1[100];
     char *pstr1;
     float x;
      int     ch;
      long    keycode, num;
     SHORT  i,j;
     SHORT  status;
     SHORT  error;

     initialize();
      if ( argc > 0)
          Print_OK = TRUE;

     for(i=0;i<3;i++) {
          nw[i].Screen = ps0;
          /* create a window */
          pw[i]= (struct Window *) OpenWindow(&nw[i]);
          if (pw[i] == NULL) {
              sprintf(buffer,"Cannot open window#%d\n",i);
             closedown(buffer);
          }
          cur_resource |= (F_WINDOW0<<i);
     rp[i] = pw[i]->RPort;
          /* establish its rastport for later */
         SetDrMd(rp[i],JAM2);
          tfont[i] = OpenDiskFont( &ttattr[i]);
          if (tfont[i] != 0) {
              error = SetFont(rp[i], tfont[i]);
              cur_resource |= (F_FONT0<<i);
          }
           /*  Attach a console to this window.  */
          sprintf(buffer,"Console %d",i);
     if ( (console[i] = AttachConsole(pw[i],buffer) ) == 0) {
            sprintf(buffer,
```

```
   "Cannot attach console #[%d] to Window #[%d].\n",i,i);
        closedown(buffer);
     }
     cur_resource |= (F_CONSOLE0<<i);
      /* Set Mode so that linefeed=LF+return. */
     sprintf(buffer,"%c[20h",27);
     conputs(buffer,console[i]);
     sprintf(buffer,"%c[0%d;40m",27, 30+nw[i].DetailPen);
     conputs(buffer,console[i]);
      sprintf(buffer,"Hello World, this is Console-%d\n",i);
     conputs(buffer,console[i]);
};
 /* End of for loop which opens windows,
     fonts and consoles.   */

 strcpy(buffer,"This is a test of conputc\n");
i=0;
do {
     ch = buffer[i];
    conputc(ch,console[0]);
    conputc(ch,console[1]);
    conputc(ch,console[2]);
    i++;
}
 while( (ch!=0) && (i<60) );

 strcpy(buffer,"This is a test of CONPUTS\n");
conputs(buffer,console[0]);
conputs(buffer,console[1]);
conputs(buffer,console[2]);
 conputs("ACTIVATE THIS WINDOW AND TYPE ANYTHING\n",console[0]);
 conputs("Use CTRL-S to change to string input mode\n",console[0]);
 conputs(" or CTRL-C to EXIT\n",console[0]);

 Func_flags |= KEYMAP_ON;
 Func_flags |= CONPUTC_ON;
j=0;
do {
     ch = congetc(console[0]);
      sprintf(buffer,"     INPUT CHAR = $%x\n",ch);
    conputs(buffer,console[0]);
    switch(ch) {
    case CR:
 /*   Output Carriage Return <CR> as Line Feed <LF>.   */
       conputc(LF,console[1]);
       conputc(LF,console[2]);
       break;
    default:
       conputc(ch,console[1]);
       conputc(ch,console[2]);
       break;
     } /*  End of switch(ch)   */
}
 while ( (ch != CTRL_C) && (ch != CTRL_S) );

 if (ch == CTRL_C)
    goto main_exit;

 conputs("Use CTRL-S <RETURN> to test\n
          numerical input modes.\n",
  console[0]);
 conputs(" or CTRL-C to EXIT\n",console[0]);
 Func_flags = Func_flags & ~CONPUTC_ON;
 Func_flags = Func_flags |  CONPUTS_ON;
while(TRUE) {
    congets(In_buf,80L,console[0]);
    conputc(LF,console[0]);
    conputs(In_buf,console[1]);
    conputc(LF,console[1]);
    conputs(In_buf,console[2]);
    conputc(LF,console[2]);
     if ( (pstrl=index(In_buf,CTRL_C)) != 0L)
        goto main_exit;
```

continued...

```
      if ( (pstrl=index(In_buf,CTRL_S)) != 0L)
         break;
   }
   while(TRUE) {
        conputs("ENTER an INTEGER NUMBER  ",console[0]);
       congets(In_buf,80L,console[0]);
        if ( (pstrl=index(In_buf,CTRL_C)) != 0L)
           goto main_exit;
        if ( (pstrl=index(In_buf,CTRL_S)) != 0L)
           break;
       sscanf( In_buf,"%d",&num);
        sprintf(buffer,"\n     num = %d\n\n",num);
       conputs(buffer,console[0]);
   }
   while(TRUE) {
        conputs("ENTER a HEXADECIMAL NUMBER  ",console[0]);
       congets(In_buf,80L,console[0]);
        if ( (pstrl=index(In_buf,CTRL_C)) != 0L)
           goto main_exit;
        if ( (pstrl=index(In_buf,CTRL_S)) != 0L)
           break;
       sscanf( In_buf,"%x",&num);
        sprintf(buffer,"\n     num = %d = $%x\n\n",num,num);
       conputs(buffer,console[0]);
   }
   while(TRUE) {
        conputs("ENTER a FLOATING POINT NUMBER  ",console[0]);
       congets(In_buf,80L,console[0]);
        if ( (pstrl=index(In_buf,CTRL_C)) != 0L)
           goto main_exit;
        if ( (pstrl=index(In_buf,CTRL_S)) != 0L)
           break;
       sscanf( In_buf,"%f",&x);
  sprintf(buffer,"\n     x = %f \n     x = %e\n\n",x,x);
       conputs(buffer,console[0]);
   }
```

```
main_exit:
     closedown("Window I/O test is finished.");

} /* END OF MAIN */

/* ATTACHCONSOLE =*/
/*   Open a console device and attach it to the indicated
window. this function returns a non-zero pointer if the
console device opened correctly and a zero (0) value if
there was an error.
 */
struct Console * AttachConsole( window, name)
struct Window *window;
char *name;
{
     static char string[40];
     int    consize, error;
     register UBYTE con_flags;
     struct Console *console;

     con_flags = 0;
     consize = sizeof(struct Console);
     if ( (console = (struct Console *)
        malloc(consize) ) == 0) {
        if (Print_OK)
            printf("In ATTACHCONSOLE,
              malloc() returned zero.\n");
        goto error_exit;
     }

      /*  Open Port for writing to console.  */
     strcpy(string,name);
     strcat(string,".write");
      if ( (console->WritePort = CreatePort(string,0L) ) == 0)
{
        if (Print_OK)
    printf("In ATTACHCONSOLE, CreatePort(%s) returned zero.\n",
                  string);
        goto error_exit;
     }
     con_flags |= CON_WPORT;
     if ( (console->WriteMsg =
       CreateStdIO(console->WritePort) ) == 0) {
        if (Print_OK)
            printf("In ATTACHCONSOLE,
             CreateStdIO(%s) returned zero.\n",
                  string);
        goto error_exit;
     }
     con_flags |= CON_WMSG;

/*   Open Port for reading from console.   */
     strcpy(string,name);
     strcat(string, ".read");
      if ( (console->ReadPort = CreatePort(string,0L) ) == 0)
{
        if (Print_OK)
    printf("In ATTACHCONSOLE, CreatePort(%s) returned zero.\n",
                  string);
        goto error_exit;
     }
     con_flags |= CON_RPORT;
     if ( (console->ReadMsg =
         CreateStdIO(console->ReadPort) ) == 0) {
        if (Print_OK)
      printf("In ATTACHCONSOLE,
      CreateStdIO(%s) returned zero.\n",
                  string);
        goto error_exit;
     }
     con_flags |= CON_RMSG;
     console->WriteMsg->io_Data = (APTR) window;
     console->WriteMsg->io_Length = sizeof(*window);
     if ( (error =
            OpenDevice("console.device",0,
             console->WriteMsg, 0) ) != 0) {
        if (Print_OK)
         printf("In ATTACHCONSOLE,
         OpenDevice returned error=%d.\n", error);
        goto error_exit;
```

```
    }
    console->ReadMsg->io_Device = console->WriteMsg->io_Device;
    console->ReadMsg->io_Unit   = console->WriteMsg->io_Unit;

    /* Point to readbuffer for the first character.  */
    console->ReadMsg->io_Command = CMD_READ;
    console->ReadMsg->io_Data = &console->readbuffer[0];
    console->ReadMsg->io_Length = 1;
    SendIO(console->ReadMsg);
    return(console);

error_exit:
    if (con_flags & CON_RMSG) DeleteStdIO(console->ReadMsg);
    if (con_flags & CON_RPORT) DeletePort( console->Read-
Port);
    if (con_flags & CON_WMSG) DeleteStdIO(console->WriteMsg);
    if (con_flags & CON_WPORT) DeletePort( console->Write-
Port);
    return(0L);
}


/* CLOSEDOWN */
closedown(s)
char *s;
{
    if (cur_resource & F_FONT0)       CloseFont(tfont[0]);
    if (cur_resource & F_FONT1)       CloseFont(tfont[1]);
    if (cur_resource & F_FONT2)       CloseFont(tfont[2]);

    if (cur_resource & F_CONSOLE0)
DetachConsole(console[0]);
    if (cur_resource & F_CONSOLE1)
DetachConsole(console[1]);
    if (cur_resource & F_CONSOLE2)
DetachConsole(console[2]);

    if (cur_resource & F_WINDOW0)     CloseWindow(pw[0]);
    if (cur_resource & F_WINDOW1)     CloseWindow(pw[1]);
    if (cur_resource & F_WINDOW2)     CloseWindow(pw[2]);

    if (cur_resource & F_SCREEN)      CloseScreen(ps0);

    if (cur_resource & F_GRAPHICS)   CloseLibrary(GfxBase);
    if (cur_resource & F_INTUITION)  CloseLibrary(IntuitionBase);
    if (cur_resource & F_DISKFONT)   CloseLibrary(DiskfontBase);
    if (cur_resource & F_DOS)          CloseLibrary(DosBase);
    puts(s);
    exit(0);
}


/* CONGETC */
congetc(console)
struct Console *console;
{
    struct IOStdReq  *msg;
    int temp;
    while( (long) (msg=(struct IOStdReg *)
        GetMsg(console->ReadPort) ) ==NULL)
            WaitPort(console->ReadPort);
    /* get the character */
    temp = console->readbuffer[0];
    temp = temp & 0xff;
    /* Ask for next one and then return this character. */
    console->ReadMsg->io_Command = CMD_READ;
    console->ReadMsg->io_Data = console->readbuffer;
    console->ReadMsg->io_Length = 1;
    SendIO(console->ReadMsg);
    conputc(temp,console);
    return(temp);
}

/* CONGETS */
congets(strbuf,nchar,console)
char *strbuf;
int  nchar;
```

continued...

```
struct Console *console;
{
    struct IOStdReq *msg;
    int ch, i=0;
    do {
        while( (long)(msg=(struct IOStdReq *)
            GetMsg(console->ReadPort) ) ==NULL)
                WaitPort(console->ReadPort);
        ch = console->readbuffer[0]; /* get the character */
        strbuf[i] = ch & 0xff;
        /*  Ask for next character.  */
        console->ReadMsg->io_Command = CMD_READ;
        console->ReadMsg->io_Data = console->readbuffer;
        console->ReadMsg->io_Length = 1;
       SendIO(console->ReadMsg);
       conputc(ch,console);
    }
     while( (ch != 13) && ((++i) < nchar-1) );
    strbuf[i]=0;
    return(strbuf);
}


/* conputc */
/*  Output a single character to a specified console.
 *  The parameter order has been
 *  chosen to parallel that of the PUTC function.
 */
conputc(character,console)
char character;
struct Console *console;
{
    console->WriteMsg->io_Command = CMD_WRITE;
    console->WriteMsg->io_Data = (APTR)&character;
    console->WriteMsg->io_Length = 1;
   DoIO(console->WriteMsg);
    /* command works because DoIO blocks until command is
     * done (otherwise pointer to the character could become
     * invalid in the meantime).
     */
    return(0);
}

/* CONPUTS */
/* Output a NULL-terminated string
 * of characters to a console.
 * The parameter order has been chosen to
 * parallel that of the FPUTS function.
 */
conputs(string, console)
char *string;
struct Console *console;
{
    console->WriteMsg->io_Command = CMD_WRITE;
    console->WriteMsg->io_Data = (APTR)string;
    /* -1 tells console to end when it sees a
        terminating zero on the string. */
    console->WriteMsg->io_Length = -1;
   DoIO(console->WriteMsg);
    return(0);
}

/* DETACHCONSOLE */
/*  Close a console device.  */
DetachConsole( console)
struct Console *console;
{
    CloseDevice(console->WriteMsg);
    DeleteStdIO(console->ReadMsg);
    DeletePort( console->ReadPort);
    DeleteStdIO(console->WriteMsg);
    DeletePort( console->WritePort);
    free(console);
}

/* INITIALIZE */
initialize()
{
    if ( (DosBase = OpenLibrary("dos.library", 0)) == NULL)
        closedown("Can't open dos.library");
    cur_resource |= F_DOS;

if ( (DiskfontBase=OpenLibrary("diskfont.library",0)) == NULL)
        closedown("Can't open diskfont.library");
    cur_resource |= F_DISKFONT;

if ( (GfxBase = OpenLibrary("graphics.library",0)) == NULL)
        closedown("Can't open graphics.library");
    cur_resource |= F_GRAPHICS;

if ( (IntuitionBase =
        OpenLibrary("intuition.library",0)) == NULL)
        closedown("Can't open intuition.library");
    cur_resource |= F_INTUITION;

    if ( (ps0 = OpenScreen(&screen0)) ==  NULL)
        closedown("Could not open the screen !");
    cur_resource |= F_SCREEN;
   init_color_map();
}
/* INIT_COLOR_MAP */
init_color_map()
{
    static short map_values[8] =
       { /*      Color             0x0RGB */
        /* 0   Dark CYAN */   0x0077 ,
        /* 1   WHITE     */   0x0FFF ,
        /* 2   RED       */   0x0C00 ,
        /* 3   GREEN     */   0x00C0 ,
        /* 4   BLUE      */   0x000C ,
        /* 5   YELLOW    */   0x0FF0 ,
        /* 6   VIOLET    */   0x0C0C ,
        /* 7   BLACK     */   0x0000 ,
    };
    LoadRGB4(&ps0->ViewPort , map_values , 8L);
}
```

•AC•

68000 Assembly Language Programming *on the Amiga™*

This month features a program that uses two libraries, include files and graphics!

by Chris Martin

Last month, we ended with a simple counting program and an explanation of how to set-up a work-disk for assembly language programming with the Commodore Macro Assembler. I also discussed assembly include files and Amiga Kernal Libraries and Devices. This month, I will present and explain a program that uses two libraries, include files and graphics! I call the program "screen" – quite simple and uneventful, but an accomplishment nevertheless. "Screen" opens an Intuition screen and draws a green line in it.

Let's start by typing the program listed on the following pages.

Boot your system with Kickstart and your Assembly Workbench Disk. Once everything has loaded, insert the assembly work-disk in drive 0, the workbench disk in drive 1 and type the following CLI command:

```
1> ed screen.asm
```

I like to name assembly programs with an ".asm" suffix. Note that the text file created here is called a "source file." When the screen clears, you can start typing the program.

There are a few simple ED commands that may be useful when typing programs. Each command is executed by typing the ESC key, then typing the command and pressing RETURN. A command may be repeated by putting a number in front of the command. Here are some examples:

x	exit and save the text
q	quit and do not save
i	insert a line
d	delete a line
bs	block start
be	block end
db	delete block
ib	insert block
4d	delete FOUR lines
3i	insert THREE lines

Assembly source files can have comments after "*" characters, but certain restrictions apply: You must either have your comments after all program code on that line, OR you must use a whole line, starting from the first space in that line. When you type this program, you may omit any text after the "*" characters – that text is just a comment.

While typing the program, MAKE SURE you type it exactly as printed. If a single TAB is missing between columns, the compilation will not work. In the actual code, a single TAB separates the label, opcode and operand fields. There are three TABs before the commands: INCLUDE, XREF and EXT_SYS.

When you have finished typing the program, exit ED with an "X" command (remember the ESC key first, though). When you are back in the CLI, assemble the program with the following:

```
1> assem screen.asm -o screen.o -c w120000
```

Remember that source files are first assembled into object (.o) files. You must then LINK the object files to executable files. The command I noted just moments ago assembled the file screen.asm to screen.o with a workspace of 120000 bytes.

When the assembler compiles a source file, it needs a workspace. The larger the source file, the larger the necessary workspace must be. The assembler defaults the workspace to 80000 bytes (which is sufficient for most programs), but with very large programs, a workspace greater than 80000 must be defined. Remember that INCLUDE files are also considered part of your program, so be sure to consider the workspace needed for programs with many INCLUDE files.

Now, the object file created by the assembler must be made executable by linking the file. Use the following CLI command to complete this necessary task:

```
1> alink screen.o to screen library amiga.lib
```

Last issue, we discussed the ".lib" library files and Amiga libraries. ALINK looks through the program for references to

continued...

library routines (for example, the "_LVO" label in the object file). By retrieving the addresses of function calls in the object file, the program is made executable.

The Program

The program begins with the INCLUDE command, to insert the include files. Next, we externally define various system calls to be used in the program. Pointers to these routines are in the "amiga.lib" file that we link to the object file to create the executable program.

We then externally define the _AbsExecBase pointer to the Exec library (also found in "amiga.lib"). When opening the graphics and intuition libraries, we need a pointer to the Exec library because the OpenLibrary routine is a part of this library.

The program is started with the "start:" label. First, we save the stack pointer in the temporary variable "savesp," defined at the end of the program. From that point, we JSR (Jump to SubRoutine) to the four sections of the program that initialize and perform various tasks. If all goes well (the libraries and the screen are opened), the program encounters an endless loop.

Opening Libraries

Last month, I listed the names of the Amiga libraries, the names of their library base names and their functions. The libraries opened in our program are the graphics and intuition libraries. Before we open a library, we need a storage area for the pointer to the library base. At the end of the program, a LONG-WORD (32 bits) is declared by the assembler directive 'ds.l'. These long words are given labels (in this case, our names for the pointers to the graphics and intuition base addresses). The linker must know the base addresses of the libraries when compiling and the program must also have a section of text to identify the library and its version.

A library is opened with the following code:

```
move.l  _AbsExecBase,a6
lea     text,a1
        <--gets the address of the text
        which identifies the library to be
        opened.
move.q  #d0,0
SYS     OpenLibrary(a6)
        <--gets the OpenLibrary routine
        from Exec library (always open),
        which reads A1 and returns the
        pointer to our library in D0.
move.l  d0,storage
        <--puts the address of the base
        into our storage.
beq     openerror
        <--If D0 is Equal to zero, then
        the library was not opened.
        Here, I jumped to a part of the program that
        returned to the normal system.)
```

The text mentioned above is a "constant" area in the following form:

```
name    dc.b  'nameoflibrary.library',0
                  <--name and version
cnop  0,2         <--blank padding
```

The first line defines the name of the library and the version (0 defines any available version). The second line has No OPeration (cnop) and leaves two blank bytes of extra "padding."

Amiga Graphics (Briefly!)

The graphics library has two different types of commands — display commands and drawing commands. Display commands initialize screens, set colors and position the display. Drawing commands perform functions such as drawing lines, writing points, filling areas and copying rectangular blocks.

Screens are a group of bits called bitmaps. A screen has a certain depth, depending upon the number of colors in use. The following are depths corresponding to the number of colors in a screen:

Depth	Number of Colors
1	2
2	4
3	8
4	16
5	32

Intuition screens follow a structure called NewScreen which looks like:

NewScreen

LeftEdge, TopEdge,	WORD	
Width, Height,	WORD	
Depth,	WORD	
Detail Pen #, (color #)	BYTE	
View Modes,	WORD	
Screen Type,	WORD	
Font,	LONG	
Title,	LONG	
Screen Gadgets,	LONG	
Pointer to CustomBitMap	LONG	(optional)

A number from one to five, defining the number of colors to be used, makes up the depth element.

View Modes is a field that defines the display type of the screen — 0 for 320x200 resolution, HIRES for 640x200, LACE for 320x400 HIRES!LACE for 640x400.

Title is a pointer to a constant area of text for a screen title.

To open a screen, you must define the elements of this structure as part of the program and you must set aside a storage area for a pointer to the screen. You must also define storage area for the RastPort and ViewPort structures.

The RastPort structure is initialized by Intuition when you open a screen. RastPort points to the screen's BitMap and keeps track of the current drawing pen, patterns, text attributes, current pen position, drawing modes and a write mask. Generally speaking, the RastPort structure is used with the graphics library's drawing routines.

The ViewPort structure controls the color palette and the View Modes, as well as the position, height and width of the display. The ViewPort is generally used with the graphics library's display routines.


```
*****************************************************************************
*
* Assembly program that opens an Intuition scrn and draws a line
*     in it.
*
* by Chris Martin
*
* When typing this program, all comments may be ommitted (anything
* that has a '*' character before it).  Also, when the program is
*        run, there is no way to exit it except:
*           CTRL - Amiga open - Amiga closed.
*
*  Compile the program with the following commands:
*
*      1> assem screen.asm -o screen.o -c w110000
*      1> alink screen.o to screen library amiga.lib
*
*****************************************************************************

        INCLUDE  "exec/types.i"
        INCLUDE  "exec/funcdef.i"
        INCLUDE  "intuition/intuition.i"

* Macros

EXT_SYS  MACRO
         XREF     _LVO\1
         ENDM

SYS      MACRO
         JSR      _LVO\1
         ENDM

* These are system routine calls used in the program.

         EXT_SYS  OpenLibrary    * (libname,version)(A1,D0)
```

continued...

```
        EXT_SYS  OpenScreen      * (newScreen) (A0)
        EXT_SYS  Move            * (rp,x,y) (A1,D0,D1)
        EXT_SYS  Draw            * (rp,x,y) (A1,D0,D1)
        EXT_SYS  SetAPen         * (rp,pennum) (A1,D0)
        EXT_SYS  SetDrMd         * (rp,drawmode) (A1,D0)
        EXT_SYS  SetRGB4         * (vp,num,r,g,b) (A0,D0,D1,D2,D3)
        EXT_SYS  SetRast         * (rp,pennum) (A1,D0)
        EXT_SYS  ShowTitle       * (Screen, 1=off or 0=on) (A0,D0)

*************************************************
*          Externally defined variables         *
*************************************************

        XREF  _AbsExecBase       * the base of the Exec library

* This is considered an External Reference because it is
*   in the file 'amiga.lib'

*************************************************
*           Start of the program                *
*************************************************

start:

   move.l   sp,savesp          * save the current stack pointer
   jsr      openlibraries      * jump to routine to open libraries
   jsr      openscreen         * open the screen
   jsr      setcolor           * set colors and clear the screen
   jsr      drawline           * draw the line

infinite:

   jmp      infinite           * this is an infinite loop

*************************************************
*             Open Libraries                    *
*************************************************

openlibraries:

* Intuition library

   move.l   _AbsExecBase,a6    * address of the Exec lib in A6
   lea.l    intuition,a1       * info about Intuition lib in a1
   moveq    #0,d0              * clear D0
   SYS      OpenLibrary(a6)    * Remember our macro?
   move.l   d0,intuitionbase   * Address of the lib put into d0
   beq      openerror          * OR,if d0 = 0, goto to openerror


* Graphics library

   move.l   _AbsExecBase,a6
   lea.l    graphics,a1
   moveq    #0,d0
   SYS      OpenLibrary(a6)
   move.l   d0,graphicsbase
   beq      openerror

   rts                         * return if all is OK

*************************************************
*             Open Screen                       *
*************************************************

openscreen:

   move.l   intuitionbase,a6   * pointer to intuition library
   lea      screenvars,a0      * attributes of the scrn in a0
   SYS      OpenScreen(a6)     * call routine
   move.l   d0,newscreenptr    * ptr to the newly made screen
   beq      openerror          * Oh Oh - open error if d0 = 0

   add.l    #sc_RastPort,d0    * get the RastPort from d0
   move.l   d0,myrp
   sub.l    #sc_RastPort,d0    * restore d0
   add.l    #sc_ViewPort,d0    * get address of ViewPort
   move.l   d0,myvp

****** Now show the screen title *****

   move.l   intuitionbase,a6
   move.l   newscreenptr,a0
   move.l   #0,d0
   SYS      ShowTitle(a6)

   rts

*************************************************
*           Set the colors                      *
*************************************************

setcolor:

   move.l   graphicsbase,a6
   move.l   myvp,a0
   move.l   #0,d0              * color number 0
   move.b   #0,d1              * RED = 0
   move.b   #0,d2              * GREEN = 0
   move.b   #0,d3              * BLUE = 0
   SYS      SetRGB4(a6)        * set the color #0

   move.l   graphicsbase,a6
   move.l   myvp,a0
   move.l   #1,d0              * color number 1
   move.b   #0,d1              * RED = 0
   move.b   #15,d2             * GREEN = 15 (highest level)
   move.b   #0,d3              * BLUE = 0
   SYS      SetRGB4(a6)        * set the color #1

* Set the drawing mode

   move.l   graphicsbase,a6
   move.l   myrp,a1
   move.b   RP_JAM1,d0
   SYS      SetDrMd(a6)

* Clear the drawing screen - SetRast to color 0 (black)

   move.l   graphicsbase,a6
   move.l   myrp,a1
   move.b   #0,d0
   SYS      SetRast(a6)

   rts

*************************************************
*           Draw a line in the screen           *
*************************************************

drawline:

* First, set the pen number

   move.l   graphicsbase,a6
   move.l   myrp,a1
   move.b   #1,d0          * color number 1
   SYS      SetAPen(a6)

* Now, MOVE to starting point and DRAW to final point.

   move.l   graphicsbase,a6
   move.l   myrp,a1
   move.w   #10,d0            * coordinates are (10,15)
   move.w   #15,d1
   SYS      Move(a6)

   move.l   graphicsbase,a6
   move.l   myrp,a1
   move.w   #300,d0            * coordinates are (300,190)
   move.w   #190,d1
   SYS      Draw(a6)
   rts

*************************************************
*             Open Error                        *
*************************************************
```

```
openerror:
   moveq    #-1,d1
   move.l   d1,d0
   move.l   savesp,sp
   rts                        * end execution of the program

*****************************************************************
*                      END  OF  PROGRAM
*
*****************************************************************

************************************************
* Library definitions                          *
************************************************

intuition   dc.b  'intuition.library',0
            cnop  0,2

graphics    dc.b  'graphics.library',0
            cnop  0,2

************************************************
* Screen title                                 *
************************************************

title       dc.b  'Assembly Language Program 1',0
            cnop  0,2

************************************************
* Uninitialized variable definitions           *
************************************************

Uninitdata     section  vars,bss

startbss

intuitionbase  ds.l  1     * this is the base address of the
intuition lib.
graphicsbase   ds.l  1     * base addr. for the graphics
library.

myrp           ds.l  1     * pointer to the screen's Rast Port
myvp           ds.l  1     * pointer to the screen's View Port

newscreenptr   ds.l  1     * pointer to the new screen

savesp         ds.l  1     * this will store the stack pointer

endbss

* Initialized variables

initdata    section  vars,data

startdata

******************** the screen's definition
************************

screenvars  dc.w  0              * LeftEdge
            dc.w  0              * TopEdge
            dc.w  320            * Width
            dc.w  200            * Height
            dc.w  2              * Depth
            dc.b  3              * Detail Pen number
            dc.b  1              * Block Pen number
            dc.w  0              * View Modes
            dc.w  WBENCHSCREEN   * Screen type
            dc.l  0              * Font
            dc.l  title          * Screen title
            dc.l  0              * Screen gadgets
            dc.l  0              * optional ptr to CustomBitMap

enddata
   end

* Always use 'end' at the end of your program.
```

•AC•

The AmigaForum on CompuServe™ Presents...

A Transcript of the

Software Publishing Conference

held on December 10th, 1986

This is a transcript of the Software Publishing conference held in AmigaForum conference room 3 on December 10th, 1986. The topic was software publishing in general, and what an unknown author can expect from a publisher in particular.

Participating as panel members were Jeff Johannigman of Electronic Arts, Brian Moriarity of Infocom, and William Volk of Aegis. Later in the evening Ben Blish of SoftCircuits joined the conference and, though not originally extended an invitation, was kind enough to join the panel and answer questions.

The following is a heavily edited transcript of that conference. Guest speaker's and moderator's comments are identified with their initials, I.E., "RR:". Forum members are designated with their names inside parenthesis, I.E., (Marlene Zenker).

RR: Greetings all, and welcome to the first official AmigaForum Conference!

The topic tonight is software publishing. How can a struggling, unknown software author improve the odds of getting his wares to market? Which publication route is best? How much return should the author expect?

We are delighted to have with us representatives from several publishing houses who will be giving us their viewpoints on these matters. Our scheduled guests are:

Jeff Johannigman from Electronic Arts
Brian Moriarity from Infocom
William Volk from Aegis Development, Inc.

These people are active on the AmigaForum, and you have most probably exchanged messages with them in the past. Tonight they have agreed to come together here, electronically, to discuss the ins and outs of publishing.

The first questions deal with the initial decision to publish.

What sort of guidelines can you suggest a programmer use to determine if his or her program is commercially viable? Jeff, you're first.

JJ: Rick, we have three criteria, and have recently added a fourth. The first three are SIMPLE, HOT, and DEEP.

SIMPLE - a good interface, no complicated learning curve, intuitive.
HOT - fully utilizes the technology, exciting, graphically interesting.
DEEP - lasting value, flexibility, replay value, etc.

And, our fourth criteria is what folks in Marketing call a "high concept"; namely, can you describe this product to me in one sentence and have me understand pretty exactly what it is.

RR: Does the "high concept" have to do with marketability, I.E., will a person reading the package understand what the program is supposed to _do_?

JJ: Yes, it is a hard lesson we learned from great programs like MULE and Worms?. They were simple, hot, and deep, but didn't sell very well because people didn't understand what they were until AFTER they had bought and played with it. On the other hand, some top ten programs sell well on the basis of high concept alone, i.e. a wrestling or karate game.

RR: An interesting point. Brian, even though your company does _not_ accept submissions, I think your thoughts on the question would still be valid. Would you like to comment?

BM: Unlike EA, Infocom does not believe there is a simple formula that can guarantee success in the software market — particularly the home entertainment market, which is as fickle and unpredictable as the record and film industries. Some of our biggest selling titles nearly didn't make it out the door, because few believed anybody would like them. We've also had our share of disappointments; eg, titles we were sure would sell really well, but whose actual performance was only mediocre. The point is that, NO ONE can predict what will and will not be popular. We can only write the best stories we can, and hope that our work captures the imagination of the distributors, dealers and buyers.

RR: This is off the track slightly, but I think this will give people a better idea of what we're looking at here. You mentioned some programs whose market performance was only "mediocre"; what is mediocre? How many pieces sold is considered a success?

BM: I can't give you exact numbers, of course; but we're very sorry when one of our titles DOESN'T "go silver." To be Silver, a title must be certified by the Software Publishers Association to have sold >50K copies.

RR: Thank you, Brian. Let's move now to William Volk of Aegis.

WV: Aegis welcomes submissions. What we look for are people with a high level of excellence. Execellence in interface, design, and ideas. I've been on both sides of the publishing/ developing issue. The best advice I can give for submissions is to look at what the publisher says and does. Aegis has concentrated on Desktop Video and Desktop Design (on the Amiga and elsewhere), products that fall into these areas are of interest to us. However anything that smacks of greatness will also catch our interest (it has happened).

We believe in the Amiga/Mac style of interface. If you product has the polish and interface of a Living Videotext MORE or Mac Paint it will get alot more attention that something that resembles a PC type product. Developers who show pride in their projects also make a lasting impression.

We like to build a group of developers that can work on many projects with us. All the Amiga products (except the Draws) were created by outsiders. So if you have something hot... let us see it. We also try to make developers comfortable, comfortable with the deal, development, and documentation. That about covers that.

RR: Thank you, William. Let's move to the second question, and I will direct this to Brian first:

What are the advantages of marketing a program through a publisher over the programmer doing it all himself? Conversely, what are the disadvantages?

Brian, even though you do not accept outside submissions, again, we would appreciate your insight.

BM: The disadvantage is that you lose control. The advantages are many. For one thing, it takes an awesome amount of money to market software nowadays, especially entertainment software. Few individuals or small companies have the clout to compete with a Broderbund, or an EA, or (dare I say it) an Infocom. And if people don't hear about your product, they won't buy it. The other big advantage is distribution. It takes years to build up an effective method of distributing product. This boring activity is best left to companies which have already made the investment in time and money. Another advantage is prestige. When a buyer walks into a store and sees a game under the EA logo, they know they can expect a certain kind of quality. Same goes for any other established company.

JJ: Gosh, thanks.

BM: I said a certain kind. Didn't say WHAT kind. Heehee.

RR: Now now, children! <Grin>

Brian, you made valid points about the advantages; I'd like to ask you about the disadvantages, though. You cited "loss of control"; doesn't this depend on the type of contract made between the parties? Related to that, what sort of control would the author expect to give up?

BM: The amount of control you get depends on who you are. If your name is Douglas Adams, and you've got a 7-million-copy bestseller under your belt,you can demand almost anything you want. If you're a nobody, you'd better beready for some marketeer to put a title and cover on your game that'll makeyou want to cry.

RR: Okay, Brian. Point well taken. William tells me he wishes to comment on the matter of "loss of control"; I will ask him to do so now, as well as to make any remarks he wishes about the original question.

WV: Control is an issue that can be part of the negotiation process. We have had developers that have retained various degrees of control of their product. This depends on what they want/need to control, the product (if it's killer, we'll talk), and the developer (maybe he knows this product's market). You could also trade $$$ for control. If by control you mean some understanding on the level and quality of advertizing, that can also be talked about.

Now to the question at hand. Some software can be self-published. I refer to highly vertical applications like developer tools or software for a very small segment of the user base (which isn't too large on the Amiga yet). However on a $$$ basis, publishing via a commerical developer can (and should) return more than self publishing (except for the cases I mentioned).You should get a fixed $$$ amount on every product sold, % of profit is meaningless since there is no thing such as profit. Most importantly you'll want your name on the package, so the next time you do a title you are already recognized.

RR: William, a comment and a question, then I will pass control to Jeff. Your comment about reaping more via a publisher surprised me; in my (admittedly uninformed) way of thinking I would have thought you would return less; this is an important point. The question: did I understand youto say "there is no such thing as profit", and what exactly did you mean?

WV: The quote "there is no thing as profit" refers to contracts where the developer gets a certain precentage of profit. The profit can dissappear with any reasonably competent accountant. Some times a company may not make any profit (for real) on a product in the first year due to marketing costs. As to the question asked. If you don't have your own publishing company and you aren't doing a small segment/high price package (ala PCLO as a good example), your publishing startup costs, and the sales you don't get because you don't

continued...

have the leverage of having been there before, and the high cost of low volume manufacturing (disks, boxes, docs etc.) will cost you enough so you will lose $$$$ compared to a good publishing contract. Assuming you can get a good publishing contract.

[PCLO is a printed circuit CAD package by SoftCircuits]

RR: Thank you for the clarification, William. Jeff, I believe you wanted to make a comment about "loss of control" and the question in general?

JJ: First, I don't know of anybody in the industry who bases their royalties on % of profit. All the ones I've dealt with count % of REVENUE. That...

WV: It died out in the early 80's... thank heavens.

JJ:... is a % of the wholesale price of each unit sold. Second, "loss of control" simply means, do you trust us? It is the job of out manual writers to write the best possible manual for your program, the job of our package designers to design the best possible package, etc. Remember, we are all working toward the same goal, to sell as many copies of your program as possible. But, by economy of scale, we have people who can specialize in each facet of getting your program to market. Admittedly, you won't agree with all of our decisions, but these are human judgments.

RR: Gentleman, let me move quickly to questions which I'm sure interest us all: MONEY. Then we will open the forum for general questions from the audience.

BM: Money? Where?

RR: Assuming you are contacted regarding a program for consideration, are you willing to sign a non-disclosure agreement? And, if you decide to accept the program, must the writer assign the copyright to you? William first.

WV: We WANT a non disclosure in order to evaluate a product. Protects the developer, protects us. Must they assign copyright? Most of the time yes... but once again, it depends on the circumstances. Often the copyright will get held by an investor group. We fund many of our projects in a similar manner as the film industry. But it could be possible for a developer to surrender marketing rights (world wide) and retain their copyright. For example the file requester program used in Draw, Draw Plus, and Diga! copyright belongs to C. Heath, though he gave us permission to use the code in our program. This happens quite often.

["C. Heath" is Charlie Heath, author of the TxEd editor and FastFonts]

RR: Thank you, William. Jeff, would you care to comment on your policy?

JJ: First question: yes, absolutely, a non-disclosure protects both of us. However, you must remember that if you submit a baseball game to us and we publish somebody else's instead, you can't really claim that that idea is yours and a breach of non-disclosure.

Second, the copyright is generally held by the AUTHOR for most of EA's programs. We have always been in favor of a software artists rights to hold the copyright, get his name (and company logo) on the front of the package, picture inside, bio notes, etc. just like records and books.

RR: Jeff, that is in fact a very visible point with EA; even in the advertising you include the programmers and references to their companies.

Brian, since you do not accept submissions, this question is not really applicable to you. Did you wish to comment in any way?

BM: I can, for amusement value, tell you what happens when we get an outside submission in the mail (which happens several times a week).

RR: Yes, please.

BM: As soon as we determine that the envelope contains a submission we seal it up without so much as GLANCING at it...

WV: Boy, that would clear my desk in a hurry!

BM:... and ship it back with a "nondisclosure agreement" which, if signed, would assign all possible rights in the known universe to Infocom, forever. So far, nobody has returned a form.

RR: Can't say as that surprises me. <Grin> Thank you, Brian. Now, the final question, the one about the big buck$, and then it's open to all.

If unsolicited material (in whatever stage of completion) is accepted by you, will you pay advances against royalties? If, so, is it possible to estimate the range of the advance? And, if accepted, what royalty percentage will your company pay to the author? If the author is not an unknown, has a track record, would the royalty percentage likely be higher?

Jeff, this one should go to you first.

JJ: Okay, I can't give exact numbers, but royalty rates in the industry have basically settled to the 10-15% range. That, of course, depends on a track record, market viability, etc. etc. We accept projects in all states, from complete bug-free (hah!) code...

RR: Hah! indeed.

JJ:... to ideas on napkins. We negotiate advances on royalties to cover living expenses and equipment costs. Figure out about how long it will take you to complete a project, and what you need to pay rent, put food on the table, etc. I can't give exact numbers, but it is a living in most cases.

RR: Okay, Jeff, thank you. Brian, since your company does not accept submissions, and since I'd like to expedite this, I'll skip you if there are no objections and go directly to William.

BM: None. Proceed.

WV: The percentage we give is from the wholesale price of the product. The numbers are the same IN MOST CASES.

We like to be involved in the development of the product, it doesn't have to be nearly complete; demos do help. It does help to be established (it can also hurt, more on that latter).

We can talk about advances. Often we get the developer hardware/software as part of the deal (Hard Drives, Aztec Compiler etc). Most developers want more than money, they want recognition and maybe some control (i.e. doing things with the product, features of the product). This isn't an absolute, we'll always listen. We have given some developers very large royalities, large advances and even more. Why? Because there software was GREAT!

continued...

Developers also feel more comfortable if the contracts are reasonable in size and understandable. We want to make deals, not break deals. I like to be able to refer developers to other developers on our ability to keep them happy.

O.K. back to the "established" developer. A product has got to stand by itself. The product counts more than almost anything else. Technology that can be used for many projects is also valuable. And since we are involved in C.D.I. (Compact Disk Interactive) publishing/development, titles that relate to this may get better deals. The bottom line is, every deal is unique.

RR: Good point! Jeff, I believe you had a comment on this?

JJ: I wanted to mention some of the other things to consider besides just the royalty points and size of advances. For instance, would it be worth 2% points to sign with somebody who can sell twice as many copies. Look at a companies sales records. Also, some of the other "perks" that come from being part of the EA Artist Community include the services of Technical Directors (like yours truly), use of our Artist Work Station Tools, ethernet access to all of our other artists, etc. (All of which is not to say we don't give great royalties <Grin>)

RR: Thanks, Jeff. The floor is now open for questions from the forum members.

John Sobernheim, you are first.

(John Sobernheim) I was wondering if any of your companies were going to market more application type tools (Integrated packages, etc.).

RR: William first, please.

WV: We are looking at integration in productivity software, can't give specific titles. But it is good if products work together in expected and un- expected ways. For example the Diga! (telecom) developers added Textronix emulation to allow Textronix images to be captured to the CAD system as an option. IFF (Thanks EA) is another good example.

A comment on the last comment. You should be concerned about the support a publisher gives to you on development. EA has been very good on this, with their developers workstation, etc. We have also worked hard to get developers the code examples they need, tools and advice. You also want to aim your product at a publisher that fits your product type. If you want to have the product sell for $499.00, you want to look for a publisher that sells $499.00 products.

RR: Thank you William. Jeff, care to respond?

JJ: Well, application tools is a large subject. Our Deluxe Series is definitely being expanded, and could potentially include word processors, page layout, sound tools, etc. We are DEFINITELY looking for more Deluxe Creativity/ Productivity tools.

BB: Hola!

WV: Here's PCLO!

RR: Thank you, Jeff. Brian, I know Infocom is primarily (entirely?) game oriented, but do you have any surprises you'd like to tell us about regarding tools?

BM: A 68000 version of our relational database, Cornerstone, now exists. We showed the Atari ST version at Comdex. If sufficient interest were shown, we might be persuaded to port it over to the Amiga.

RR: Thank you, Brian. Folks, you heard the man... send mail!

John Sobernheim, you regestered an additional comment?

(John Sobernheim) SMART Plus on an Amiga or ST would be nice. Build in WP/SP/GR/Comm, and sell millions. Thats all.

RR: Okay, I'm sure your comment has been noted. William, you had a comment?

WV: On the Amiga vs. ST market for software. SPA (Software Publishers Association) survay shows that even though there are more ST's than Amiga's, more software (by volume) is selling on Amiga and by $$$ it's a larger margin. This may change. Just thought I'd give you some good news.

RR: <Grin> I'm sure that's something we are all glad to hear.

Jeff, you had an additional comment?

JJ: Actually, according to Atari, 150,000 or so ST's have been sold, but 60% of them are in Europe. That means domestically, there are more Amigas.

RR: Thanks for that, Jeff. Brian, you also had a comment.

BM: In brief, our figures suggest that Amiga buyers tend to be first-time computer buyers, wheras ST buyers are upgrading from their 8-bits. We see more of our standard classics (ZORK, etc.) sold on Amiga than on ST, possibly because more ST owners have already played them.

RR: Thanks Brian.

RR: Folks, I don't mean to put the man on the spot, but lurking in the audience is Ben Blish of SoftCircuits, makers of PCLO...

WV: Yea Ben!

RR:... you may wish to direct some questions to him (though he is under NO pressure to answer, since he is just a spectator.)

BB: I'd answer, I suppose.

RR: Glenn Nielsen, you were next.

(Glenn Nielsen) Thanks, For the small publisher or individual, how effective is word of mouth for marketing, i.e. BBS/network annoouncements and DemoWare? I'd like a comment from Ben also if he's willing.

RR: Thanx. An _excellent_ question, Glenn. Before I ask for replies I have been informed that Brian Fargo of Interplay (Bard's Tale!) is also in the audience. <visualizing dozens of heads turning <grin>>

(Brian (Interplay)) Hello all.

RR: Since Ben's just arrived, let's put him on the spot first.

BB: Well, word of mouth, not very... BBS/network, nominally better, demoware, very. IF, and this is a big if, your demoware doesn't obsolete or potentially replace your ware-ware. No save, or some equivalent cripple needs to be placed in the Software.

You should keep in mind that my company's product is aimed at a very vertical market, that is to say, one that is specialized, and so that skews my results from the general market type of stuff. On the other hand there are a lot of technical types using and buying Amiga's for the first time, and so mebbe that skews it back.

RR: Thanks, Ben. Jeff, want to field this one?

JJ: Well, word of mouth, user groups, etc. is a very tricky channel. They also tend to be the people who believe in "freeware", (i.e. sharing copies of anything they like). It only works if your end user is very ethical and/or needs follow-up support.

RR: Jeff, I think we should make the point that the classical definition of "freeware" does _not_ include pirated software, correct?

WV: Yep.

JJ: True, but there are plenty of folks who believe ALL software should be freeware.

RR: Sad (in the case of piracy), but true. Brian, care to comment?

BM: Since our products appeal to a fairly narrow, and intensely loyal, group of people, word of mouth is very important to us. My last title, TRINITY, was literally saved by it.

continued...

RR: Interesting how different circumstances result in different outlooks on some of these issues.

Peter Hodgins, you have a comment on this issue?

(Peter Hodgins) What kind of copy protection schemes do you forsee in the future?

WV: Optical Media?

JJ: Lower prices and ethical users.

(Brian (Interplay)) Off line protection.

WV: Documentation and added value.

RR: William, you wanted to comment on word of mouth.

WV: O.K., word of mouth and demos and BBS can help a vertical product get out there. The problem is that dealers may not carry the product if they don't know you or if they don't see ADs for the product. Early in a machine's history shareware (demos also) can work out for these vertical products. On the Amiga TxEd comes to mind.

A demo can come back to haunt you; if it's not as good as the released product it may be hard to get the word out. The BEST example of a demo working miracles is the Mindscape Cinemaware demo of Defender of the Crown (actually the demo was an Animation in Aegis Animator).

WV: The product was sold, marketed, and broke records because of the demand the demo built up at user group meetings and shows. It also went to reviewers as well. They did do some good advertizing but the demo sure helped. By the way, the game actually used a animation-compiler to do much of the play! (Defender of the Crown).

RR: William, I think you indirectly touched on an important point when you said "early in a machine's history". I think we all remember a certain terminal program (I won't mention the name, please, no one else either!) that we bought simply because that was all there was. Now that the system is more established, getting buyers is going to become trickier, and this may be where the publishers come into play. Am I on base?

WV: Lameware does sell for a while, but it will backfire on the publisher.

RR: Well, my point was not necessarily that the program was "lameware"; I think the same principle would apply if it was fantastic. The "only cow gets milked", as it were.

WV: Oh, the best way to break in the business is early in the history of the machine... witness Aegis.

JJ: A mature market (like the c64) relies much more on distribution channels such as the Wherehouse or Toys R Us, and on magazine ads.

RR: Folks, I want to interrupt briefly to tell you that Brian Moriarity of Infocom needs to slip out. I'm sure we all appreciate his being here, and if this were face to face I'd call for a round of applause (with no doubt I'd get it). Brian, thank you for sharing your thoughts with us.

BM: Thanks for inviting me in. Slipping out now.

RR: I think we've beaten that question to death... Marlene, you were next.

(marlene zenker) How do you determine the price of a package and how would the formula vary if you were an individual marketing on your own?

RR: William, care to take that one first?

WV: The price depends on what the competition is selling and what packages like it sell for in general. Develop-

ment costs also play a role. However I've situations where developers attempted to price product on the basis of "You would have to spend $20,000 on a CubiComp to do this", it doesn't work that way. You can't really charge on a basis of precieved value, you have to fit the marketplace.

RR: Thank you. Jeff, you had a comment?

JJ: Yes, laws of supply and demand tend to drive the prices as low as possible. Take our c64 market, for instance. Now that it is so large, we can afford

[Long pause]

RR: Jeff, did we lose you?

[Long pause]

Looks like Jeff's modem is playing games with him. Ben, you had a comment about this issue in the meantime?

WV: Ben, I wasn't referring to anything like PCLO, your price is fair.

BB: Yeah, if there is NO competition (our case when we started), then your price can be set to whatever the market will bear in exactly the manner suggested, IE it would cost XXXX to do this on a frump, so this costs .5*XXXX

WV: But not for a potential mass market item (no names).

BB: If there is competition, then that sets the price all by itself, unless your product is so superior that it cannot be missed or spoken about in a misleading way. Bill's too smart to make those kind of claims; others aren't.

And, it hasn't seemed to make a lot of difference to us that there is a company attempting to compete with us, lower price, same target market. So even if you are competing, use excellence as a lever and you won't have to worry so much about the price.

JJ: I'm back, don't know what happened.

(marlene zenker) So basically it doesn't matter whether you're an individual or large publisher, the marketplace is what counts.

BB: I don't think so. I'm an individual, and thats all there is to SoftCircuits.

WV: And we refer folks to Ben all the time!

RR: Well, William has just told me he must be moving out. Thanks very much for coming tonight!

WV: Any final querys?

RR: If you have, time, William, Steve our Wizop had one very important one which I think should be asked.

WV: O.K.

RR: Steve, are you available?

[Long pause]

Apparently not, but let me ask it, because I believe it IS important: "Do you regret your decision to develop for the Amiga, and what do you see in it's future?"

William, go ahead first, since you need to leave.

WV: Are you kidding! The Amiga really moved Aegis up! However I wish Commodore would do more for the marketplace and promote software that "followed the rules" etc. etc.

(John Sobernheim) Hear Hear!!

WV: No regrets, we have ALOT more Amiga product in the works.

continued...

RR: Thanks, William; I doubt anyone expected to hear anything else. Thanks for spending time with us tonight. Have a good evening (what's left of it!)

WV: Bye for real.

RR: Jeff, would you like to respond to Steve's question, please.

JJ: Absotively, posilutely, NO REGRETS! We have grown over 100% this year, and a lot of it is due to our early commitment to the Amiga, especially with the Deluxe Series. We made back our initial investment in Amiga development in the first three weeks that we shipped our first Amiga releases. Our slogan used to be "We See Farther"... now, thanks to the Amiga, Farther is here. (yeah, I know that sounds corny. Sorry, I get carried away!)

(marlene zenker) Not corny, EA is a really classy company.

RR: Jeff, as long as you believe it (and I think you do), I think we can all excuse the corn. Ben, would you care to respond to this as well?

BB: Ok first, I don't regret it. Sometimes I'm a little jealous of the market share that companies get when they support popular toys like the IBM PC...

RR: <cringing, hoping there are no IBMers in the crowd>

BB:... but the Amiga seems to be still selling in spite of Commodore. When the Sidecar is released in the US I'll be a LOT happier. A company that keeps claiming, then doesn't deliver, gets the users very annoyed. I think the boys and girls at CBM are being really shortsighted, over and over again. We'll stick, though, as the machine is fabulous, and what I want out of this is performance products, at the level I, and my customers, see fit.

RR: Ben, I think the frustration felt towards the way Commodore is handling matters is almost universal, and I'm sure there are many of us who feel that WE could do a much better job of management. But I would stress one thing, something I stressed way back at the beginning of the Amiga: they have problems, true, but let's remember all the things they did RIGHT.

BB: They bought Amiga. That's one. What else?

RR: Ben, that's a pretty important one, wouldn't you say?

BB: yes, very. but it's still one.

RR: Ben, don't forget, CBM also converted the Ami to 3.5 inch drives and more memory.

[The Amiga was originally designed with 5 1/4" drives and considerably less memory]

Jeff, you had a comment on this?

JJ: Actually, I have heard a lot lately that leads me to believe that CBM is FINALLY getting its act together. RJ Mical is even getting optimistic about CBM's directions.

[RJ Mical was instrumental in the development of the Amiga's hardware, as well as being very visible in a developer support capacity. He was one of the first people let go in the Commodore layoffs earlier in the year]

RR: Okay Jeff. Okay, folks, This officially ends our first conference; I hope it was useful to you. I will now turn everyone loose to chat.

(S. Ahlstrom/SYSOP) I would VERY much like to thank our guests and Rick Rae for spending a Wednesday evening enlightening us all.

BB: Can we fight now?

•AC•

About Online Conferencing

by Richard Rae

Reading a conference transcript (either here in *Amazing Computing* or electronically via modem) is only half the story. With a transcript you can read what happened; by participating in a conference you can be a part of it. Instead of merely hoping the featured guest addresses an important issue, you can be sure, by asking a question or making a comment about it.

To participate in a conference, you must first be a member of the appropriate network. This conference was held on CompuServe's AmigaForum, a meeting place for literally thousands of Amiga enthusiasts. Joining CompuServe is as simple as following the instructions in a "SnaPak," a tear-apart envelope which contains a temporary user ID and password. Many modems and telecommunications programs include SnaPaks or you can buy a starter kit complete with SnaPak and user's guide from many electronic and computer stores. If you can't find them locally, you can order one directly from CompuServe for $39.95 by calling 1-800-848-8199 (in Ohio or Canada call 1-614-457-0802). This starter kit also includes a $25 usage credit, so the actual cost is about $15.

The next requirement is to be a member of the forum which supports your area of interest. There are scores of forums on CompuServe, each addressing a particular subject. In this case, typing GO AMIGAFORUM at any "!" prompt will take you to the AmigaForum's Visitor's Menu; simply select the "join" option to become a member. There are no special forum dues or responsibilities beyond following the general conduct rules.

Once you are a member of a particular forum, you need to know when the special conferences are scheduled. The AmigaForum normally posts a short bulletin (which you will receive automatically when entering the forum) a week in advance of each formal conference, so you need only jot down the day and time for any conference which interests you.

Of course, you'll also need to know how to attend the conference! The AmigaForum is conceptually divided into three areas: the Message Base (where members swap messages back and forth, not unlike an electronic bulletin board), the Data Libraries (where all the programs and files are stored), and the Conference Area. (All the forums on Compuserve are structured similarly, so if you learn one, you've learned them all.)

Upon entering the AmigaForum, you are in a "lobby" of sorts, with direct access to the Message Base. To move to the Conference Area, simply select the conference option from the menu or type CO at the main forum prompt.

The Conference Area is composed of several "rooms," each of which can contain a seperate conference, all going on simultaneously. When you enter, you will be in Room 18, which is the default. Informal gatherings are normally held in this room.

To participate in a formal conference, you need to go to the appropriate room. Most AmigaForum conferences with featured guests are held in Room 2, "Talk to the Trade." To get there from Room 18, type /ROOM 2 or /TUNE 2. The slash is a delimiter and should be the first character on the line for any command. For example, /EXIT will return you to the "lobby", /HELP will dislay a list of conference commands, and so on.

Formal conferences include a moderator (who acts as "traffic cop"), one or more featured guests, and the forum membership. In order to maintain a smooth flow of information, a simple protocol is followed by all participants; this protocol is explained in a text file in the forum. Again, information about this file is included in the short bulletin you will automatically receive if a conference is scheduled.

Participating in a conference (and a forum) is an exciting way to meet others with similar interests, especially industry figures with whom you'd not normally have the opportunity to talk. It is also a valuable resource. Someone usually has a rapid answer for just about any question or problem you might have. The AmigaForum works because of its members, who are bright, vocal and intensely interested in everything Amiga. Come join us!

•AC•

O to 60 in 3 Seconds

Actually, we're being conservative. The ANIM feature in VideoScape 3-D can play up to sixty frames in *one* second. Real time. Perfect for desktop video production. Perfect for desktop presentation. The ultimate 3-D animation system for the Amiga.

VideoScape 3-D has been designed to work with any Amiga computer using a minimum of 512K RAM. It features solid object generation with hidden surface removal, diffuse reflection from a light source, specular reflection, and a wire frame mode.

VideoScape's Easy Geometry Generator lets you create simple geometric shapes like cubes, spheres, boxes, and cones. You can also use Designer 3-D's visual interface to create unusual shapes. VideoScape 3-D includes a series of objects created by Allen Hastings, as well as IFF foregrounds and backgrounds painted by Jim Sachs and Richard LaBarre. You can generate frames and automatically play them back from scripts, step through each frame one at a time, and use manual or automatic camera motions. VideoScape 3-D will work in multiple resolutions up to 704 x 440 including overscan and interlace.

Combined with other software products, including Aegis VideoTitler, Aegis Animator, Images, Deluxe Paint II, or Aegis Animation Workshop, your animations will shift into high gear. Aegis puts you in the winner's circle! Join the team today!

For more information
or your nearest dealer:
(213) 392-9972

To order direct:
1-800-345-9871

AEGIS
DEVELOPMENT
2210 Wilshire Blvd., #277
Santa Monica, CA 90403

The AMICUS Network

by John Foust

As you can probably tell, I am becoming more and more fascinated with Amiga video and animation programs. This affection is appropriate for someone in love with this machine — graphics and video are the Amiga's forte.

The SIGGRAPH conference in Anaheim, California in late August was proof that the Amiga is a contender in the animation and video market. SIGGRAPH is the annual convention of members of the graphics special interest group of the Association for Computing Machinery. Most ACM members are from the best computer graphics companies and university research groups.

Beyond the technical talks, there was also a large trade-show style exhibit hall. The Commodore booth was situated near Apple and Sun Microsystems, the other "low cost" computer systems (The cheapest Sun system is priced about $10,000. A Mac II system prices out at about $6,000. The Atari ST was nowhere to be found.).

This display is no casual trade show. It is a time when the best minds in computer graphics present papers and lectures. Films and artwork are also presented and competitively judged in a competition. SIGGRAPH attendees can view the films in evening showings.

The hardware is tops, too. How good? At the film show, I was in the tenth row away from a huge 40-foot projection screen. While watching the best film of the night, I blinked to clear my eyes and held my head steady in my hands, mouth agape. I couldn't see anything "wrong" with the animation. It simply looked real – no jaggies, no hard mechanical edges, all realistic motion.

I should clarify. The word "contender" implies competition and a chance to win. The competition at SIGGRAPH were giants compared to the Amiga. After all, can a $2000 computer compete against $50,000 computers? Perhaps "competition" wasn't the right word choice. There was one Amiga-generated animation in the film show called "Dance of the Tumblers" by Steve Segal of West Hollywood, California. Done with Aegis Animator and enlarged on that 40-foot screen, the low-resolution Amiga pixels looked huge and the animation looked coarse. There was a big gap between the Amiga and the other graphics computers at the show.

The conference made me think, "What makes the Amiga great?" The answer was simple. First — low cost. After all, people who work on $200,000 computer systems by day want a computer for home, too. . . or perhaps they can't get access to the expensive computer in the office, so they can put Amigas on their desks. The low cost of the Amiga is very attractive and I am sure the Amiga 500 was the lowest cost computer shown at SIGGRAPH.

Second, the Amiga is interactive. On personal computers, the user interface is primary to the usefulness of the program. Many of these more expensive computers have inferior or non-existent user interfaces. In many cases, you are provided with documentation that assists you in programming the hardware, but little more than that. A large banner at the Pixar booth advertised a base system for $49,000 – "Including a C Compiler."

During the guided press tour, an ACM official called the Amiga the "great sleeper" of the show, saying its graphics potential was yet to be realized. I agree. I hope that many other SIGGRAPH attendees took that idea home with them, too.

Commodore in force

It was very nice to see Commodore officials present at the show. Al Duncan, Commodore's general manager, was present and on the floor every day of the show. Henri Rubin, the executive VP and chief operating officer, and Richard McIntyre, the senior VP of sales and marketing were present for at least one day of the show. I think the Commodore reps were pleased by the response from attendees; the booth was always busy. At an Amiga user group meeting during the week of the show, Duncan announced that Commodore will also be present at the next SIGGRAPH.

The week before SIGGRAPH marked the shipment of both Byte-by-Byte's Sculpt 3D and Aegis Development's VideoScape 3D. Ray- tracing was a hot

continued...

SIGGRAPH

More Animation Products

Live! Update

topic both in and out of Amiga booth. The Sculpt product raised the eyebrows of many casual observers who used much more expensive computers to do the same thing. VideoScape impressed many people for the same reasons. There were a half-dozen animations shown by people who had bought the program only days before. Aegis Development had four spots showing their line of software. Sonix provided the background music for the booth.

Mimetics showed their real-time video digitizer and frame buffer which went into production in August and should be available in September. They also showed software that can display an IFF picture on the frame buffer. The software is somewhat slow; it takes several minutes to translate the IFF data to a form suitable for the buffer. The buffer does not accept simple red, green, blue values for each pixel. The board uses a minimal amount of memory to store the image, so it must convert the pixel values to another form. This process takes time and I imagine this delay will hamper interactive applications of the frame buffer. Time problems aside, the images were superb. Videophiles were impressed by the buffers' ability to capture two true video fields. Captured images looked like very good still-frames on a video recorder.

As usual, New Tek showed their Digi-View video digitizer and their new HAM paint program, Digi-Paint. Other booth regulars included CSA with their 68020 / 68881 Turbo Amiga, Liquid Light and their Polaroid Palette screen camera, Software Visions with Micro-Fiche Filer and CalComp with their PlotMaster color printer.

Gold Disk showed Professional Page, Pagesetter and Laser Type. Professional Page is expected to ship in September. Crystal Rose showed a new fractal-based graphics program. If you remember, they showed a Mandelbrot program at last fall's Commodore Show. Anakin Research showed a version of their Easyl drawing tablet interface for the Amiga 500 and 2000. The new boards use a custom ASIC chip, so the board parts count is reduced.

Ameristar used their network boards and software to join two Amiga 2000s with a Sun minicomputer. The Sun with its 300-or-so megabyte hard disk became accessible to the two Amigas as a logical device, such as "NET:". In other words, the resources of each computer on the network are available to all. This connection is called peer-to-peer networking. Ameristar also has remote logins working over TCP/IP which were demonstrated to Sun Microsystems representatives in a private meeting.

Mindware showed an IFF animation program called PageFlipper. It loads IFF images into memory and flips them to the screen in succession. Overscan, as well as scripting of the animation sequence, is supported. One interesting feature is a seamless, looping background image for a character walking on the screen, available for $49.95. Mindware occupies the same address as Anakin Research.

The University of Lowell, Massachusetts showed a DMA image processing board for the Amiga 2000. The board can accommodate up to seven parallel image processing chips (the NAC 7281). Each chip gives approximately 5 MIPS (million instructions per second) of processing power, dedicated to image processing. The demonstration in the booth showed the board rotating a two-dimensional image, counting vertical pixels in columns and drawing histograms of the data in real time.

The Amiga was present in a few booths outside the Commodore booth. Byte-by-Byte had their own booth and a presence in the Commodore booth. Micro Magic, the makers of Forms in Flight, had a small booth near the Commodore booth. According to David Youlton, Forms in Flight was first shipped in late June. Its animation resembles VideoScape, but Forms in Flight has a very nice, interactive user interface for creating objects, while VideoScape lags in this respect. The program also works in stereo, with red-and-blue glasses. You can edit and view objects in stereo wire frame. Forms in Flight does not conform to any IFF standard, unfortunately, and uses its own format for animation files. It is also geared to users with the admittedly expensive step-frame recorder. Youlton also plans to release a freely-distributable animation player called Fast Flight.

During the first two days of the show, the General Electric booth showed the Winner's Circle Systems video presentation system. GE bundles Amiga hardware and software together for sale through Winner's Circle in Berkeley, California. In the GE booth, a seven megabyte Amiga played a 1300 frame VideoScape animation, which is all stored in memory and played back in real time. Reportedly, GE thought

they'd stump viewers by saying they couldn't use a videotape of an animation, so they did it all from memory instead.

The 1988 SIGGRAPH will be held August 1 to 6 in Atlanta, Georgia – and Commodore will be there.

Amiga Friends

The Amiga Friends, a local user group, hosted a meeting to coincide with the SIGGRAPH conference. Attendees filled a large ballroom in a hotel on the Anaheim convention grounds. It is nice to see local user groups taking advantage of the Amiga developer presence when a trade show comes to town. As someone remarked at the meeting, any competitor of Commodore could bomb this ballroom if they wanted to ruin the Amiga.

Al Duncan and Dave Archambault of Commodore spoke at the meeting. They made a few nice announcements: Commodore will have a booth at the Fall COMDEX in Las Vegas in November. They plan to continue the advertising campaign for the Amiga 500 and 2000 into the fall. Commodore has added seven new telephone customer support people, partially due to the 5000 calls a day they are still receiving concerning the Commodore 64. Commodore is also creating a plan to court Amiga 500 sales in the educational market.

The meeting featured numerous demonstrations of Amiga products by representatives of exhibiting companies. A summary of speakers and demonstrations:

CSA representatives Al Riker and Bill Reed showed their Turbo Amiga speed enhancement boards. A clear demonstration of the speed increase was given with a Mandelbrot program, well-known for slowness and an appetite for CPU cycles. New iterations of the Mandelbrot set took only seconds, while an ordinary Amiga might take minutes to do the same work. Riker and Reed hinted that CSA might be working on higher resolution video boards for the Amiga.

Micro Magic's Dave Youlton showed Forms in Flight, his shaded polygon animation package. NewTek president Tim Jenison showed Digi-Paint. Byte by Byte president Scott Peterson demonstrated Sculpt 3-D.

MicroIllusions rep Jim Steinert showed Photon Video, an upcoming line of animation programs. The demo looked as good as ray-traced animation, but used a different, much faster technique than ray-tracing. MicroIllusions is planning interfaces for controlling professional video equipment, such as a

continued...

SMPTE interface. At this point, it all looks too early to tell — they don't plan to release the product for months.

Contact: Probe

Keith Doyle showed his Contact:Probe animation. This demo disk has been making the rounds; perhaps you've seen it. Doyle participated in Contact, an annual meeting of science fiction writers and artists. They get together to talk about an imagined contact with an alien species. Each group keeps the others in check and all work together to produce stories and artwork to illustrate their conclusions about this alien contact.

Doyle's demo, produced with his partner Joel Hagen, was used in the television coverage of the Contact conference (for an hour-long show on California public television). Using a series of custom software tools, the demo animates a sequence of the telemetry returning from an interstellar probe that discovered a skull on another planet. The on-board computer fleshes out the skull and animates a bust of the alien.

Hagen hopes to turn his tools into a product, scheduled for an October release from his company, the Right Answers Group. His projected program, called The Director, uses a script language to drive the animation and the script commands support IFF and blitter operations directly. Hagen's work should make for an interesting Amiga-specific product.

Aegis Development's Bill Volk showed a video tape of animations from Videoscape 3D and talked about the terminal program Diga! and some other forthcoming Aegis products. Apparently, new versions of Aegis Draw will have "software slots," similar to the user-customizable terminal emulations in Diga! (User-customizable if you are a C programmer, that is...).

Amiga artist Jim Sachs was scheduled for a presentation of his latest work, but Volk went into overtime, and Sachs was left in the dark. The meeting consumed four full hours. Even for the most rabid Amigan, these demonstrations can be a bit tedious if the speaker is not accustomed to speaking to groups. As one demonstrator talked about "his personal favorite menu option," I overheard a heckler say something about "his personal favorite menu option, called Quit."

The program ended with a panel of Amiga experts and luminaries, including most of the previous speakers, graphics hackmaster Leo Schwab (in full cape and hat, of course) and BIX SYSOP Joanne Dow. An open question session was followed by predictions from each panel member. Luckily, I had talked with the Bandito before the meeting, so I had a series of predictions "guaranteed to come true." Former Amiga Los Gatos wizard Dale Luck had some interesting predictions about enhanced graphics chips with more CHIP memory and more colors.

New Prism

The Prism HAM picture editor, reviewed in this summer's video issue, has been updated to version 1.2. The upgrade is free for all registered owners, according to Stan Kalisher of Impulse, Inc. You can contact Impulse for more details at (800) 328–0184.

The most notable improvement in the program is how it draws. While some programs (such as Deluxe Paint) can't quite keep up with the fastest mouse movements, the new Prism does very well at reproducing the fine movements within a fast mouse movement. A signature is a formidable test. Most programs in the free-line drawing mode can't keep up with even an average signature, using the mouse like a pen. Prism has also added cleanup to the HAM images, so the HAM pictures are much sharper than before. Impulse also plans a mid-September release for a ray-tracing animation program called Silver. Silver will sell for $169 alone or $199 bundled with Prism 1.2.

Amiga Live! update

"It was for the best. Grab was not a corporation made in heaven. Grab and A Squared had different intentions for the corporation's plans for the future. Those differences proved to be too great."

RJ Mical explained the latest twist to the story of the Amiga Live! video digitizer in this manner. Live! is produced by A Squared Systems of Oakland, California. Several months ago, Mical and his wife Caryn formed a corporation called Grab, Inc. to market Live!. The corporation has now been disbanded. According to Mical, Grab did not accept any prepayments for Live!, so there are no unhappy customers.

Because of differences of opinion in marketing strategy, Grab and A Squared agreed to dissolve the corporation on July 30.

A Squared still retains the rights of distribution of Live!. They have formed another company to distribute through both direct mail-order sales and computer dealers. According to Wendy Peterson, president of A Squared Systems, the planned September 15 ship date still holds, along with the $295 list price.

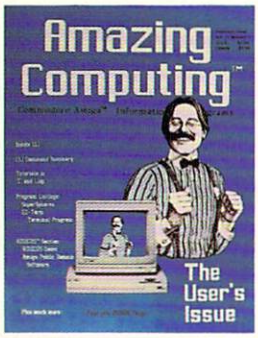
"It's simply a name change, that's the best way to describe it," she said. "The phone numbers are the same. The address is the same." Peterson and Mical agree on the reason for the breakup. She said, "We formed a company and found that we had ideas that were going in different directions."

More shows

Two exclusive Amiga shows happen in October. The first is the Commodore Show in Anaheim, California, on October 3 and 4. The AmiExpo takes place in New York the following weekend, October 11 to 12. Amazing Computing™ will have three booths at AmiExpo. The booth will be staffed during the show by Many AC™ authors. AmiExpo is the first Amiga-only show on the East Coast.

•AC•

Expanding Reference

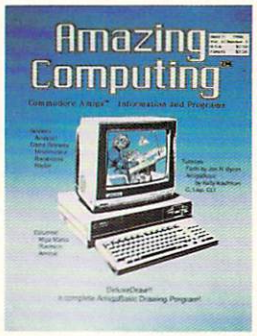


Volume 1 Number 1 Premiere 1986

Super Spheres By Kelly Kauffman An ABasic Graphics prog.
Data Virus By J Foust A disease may attack your Amiga!
EZ-Term by Kelly Kauffman An ABasic Terminal program
Miga Mania by P. Kivolowitz Programming fixes & mouse care
Inside CLI by G. Musser a guided insight into the AmigaDos™
CLI Summary by G. Musser Jr. A list of CLI commands
AmigaForum by B. Lubkin Visit Compuserve's Amiga SIG
Commodore Amiga Development Program by D. Hicks
Amiga Products A listing of present and expected products

Volume 1 Number 2 March 1986

Electronic Arts Comes Through A review of software from EA
Inside CLI: part two G. Musser investigates CLI & ED
A Summary of ED Commands
Live! by Rich Miner A review of the Beta version of Live!
Online and the CTS Fabite 2424 ADH Modem by J. Foust
Superterm V 1.0 By K. Kauffman A term. prog. in Amiga Basic
A Workbench "More" Program by Rick Wirch
Amiga BBS numbers



Volume 1 Number 3 April 1986

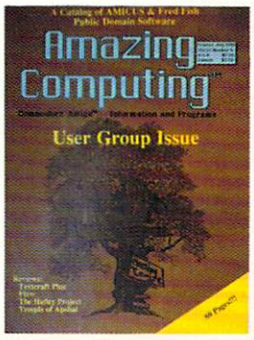
Analyze! a review by Ernest Viveiros
Reviews of Racter, Baratacca and Mindshadow
Forth! The first of our on-going tutorial
Deluxe Draw!! by R. Wirch An Amiga Basic art program
Amiga Basic, A beginners tutorial
Inside CLI: part 3 by George Musser George gives us PIPE

Volume 1 Number 4 May 1986

SkyFox and Arcticfox Reviewed
Build your own 5 1/4 Drive Connector By Ernest Viveiros
Amiga Basic Tips by Rich Wirch
Scrimper Part One by P. Kivolowitz prog to print Amiga screen
Microsoft CD ROM Conference by Jim O'Keane
Amiga BBS Numbers

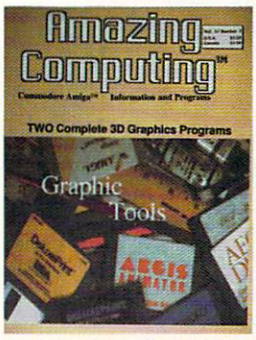
Volume 1 Number 5 1986

The HSI to RGB Conversion Tool
by S. Pietrowicz Color manipulation in BASIC
AmigaNotes by Rick Rae The first of the Amiga music columns
Sidecar A First Look by John Foust A first "under the hood"
John Foust Talks with R. J. Mical at COMDEX™
How does Sidecar affect the Transformer
an interview with Douglas Wyman of Simile
The Commodore Layoffs by J. Foust A look Commodore "cuts"
Scrimper Part Two by Perry Kivolowitz
Marauder reviewed by Rick Wirch
Building Tools by Daniel Kary



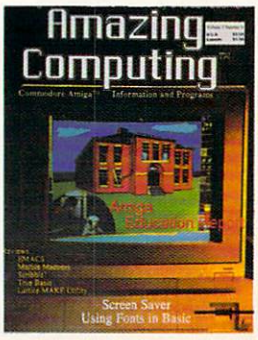
Volume 1 Number 6 1986

Temple of Apshai Trilogy reviewed by Stephen Pietrowicz
The Hailey Project: A Mission in our Solar System
reviewed by Stephen Pietrowicz
Flow: reviewed by Erv Bobo
Textcraft Plus a First Look by Joe Lowery
How to start your own Amiga User Group by William Simpson
Amiga User Groups
Mailing List by Kelly Kauffman a basic mail list program
Pointer Image Editor by Stephen Pietrowicz
Scrimper: part three by Perry Kivolowitz
Fun With the Amiga Disk Controller by Thom Sterling
Optimize Your AmigaBasic Programs for Speed by Pietrowicz



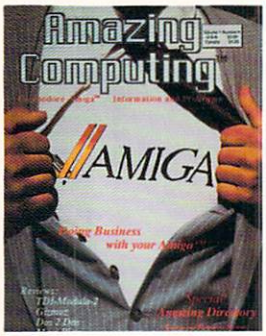
Volume 1 Number 7 1986

Aegis Draw: CAD comes to the Amiga by Kelly Adams
Try 3D by Jim Meadows an introduction to 3D graphics
Aegis Images/ Animator: a review by Erv Bobo
Deluxe Video Construction Set reviewed by Joe Lowery
Window requesters in Amiga Basic by Steve Michel
ROT by Colin French a 3D graphics editor
"I C What I Think" Ron Peterson with a few C graphic progs
Your Menu Sir! by B Catley program Amiga Basic menues
IFF Brush to AmigaBasic 'BOB' Basic editor by M Swinger
Linking C Programs with Assembler Routines on the Amiga
by Gerald Hull



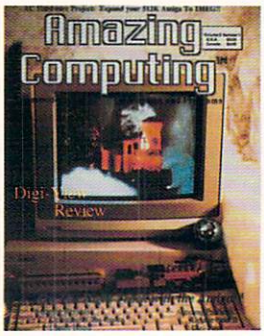
Volume 1 Number 8 1986

The University Amiga By G.Gamble
Amiga's inroads at Washington State University
MicroEd a look at a one man army for the Amiga
MicroEd, The Lewis and Clark Expedition reviewed Frizelle
Scribble Version 2.0 a review
Computers in the Classroom by Robert Frizelle
Two for Study by Frizelle Discovery & TheTalking Coloring Book
True Basic reviewed by Brad Grier
Using your printer with the Amiga
Marble Madness reviewed by Stephen Pietrowicz
Using Fonts from AmigaBasic by Tim Jones
Screen SaVer by P. Kivolowitz A monitor protection prog. in C
Lattice MAKE Utility reviewed by Scott P. Evernden
A Tale of Three EMACS by Steve Poling
.bmap File Reader in Amiga Basic by T Jones



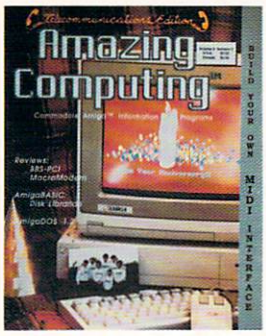
Volume 1 Number 9 1986

Instant Music Reviewed by Steve Pietrowicz
Mindwalker Reviewed by Richard Knepper
The Alegra Memory Board Reviewed by Rich Wirch
TxEd Reviewed by Jan and Cliff Kent
Amazing Directory A guide to the sources and resources
Amiga Developers A listing of Suppliers and Developers
Public Domain Catalog A listing of Amicus and Fred Fish PDS
Dos 2 Dos review R. Knepper
Transfer files from PC/MS-DOS and AmigaBasic
MaxiPlan review by Richard Knepper The Amiga Spreadsheet
Gizmoz by reviewed by Peter Wayner Amiga extras!
The Loan Information Program by Brian Catley
basic prog. to for your financial options
Starting Your Own Amiga Related Business by W. Simpson
Keep Track of Your Business Usage for Taxes by J. Kummer
The Absoft Amiga Fortran Compiler reviewed by R A. Reale
Using Fonts from AmigaBasic, Part Two by Tim Jones
68000 Macros on the Amiga by G. Hull Advance your ability.
TDI Modia-2 Amiga Compiler review by S Faiwsze



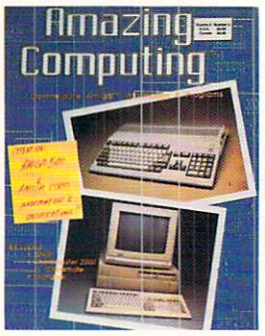
Volume 2 Number 1 1987

What Digi-View Is... Or, What Genlock Should Be! by J. Foust
AmigaBasic Default Colors by Bryan Catley
AmigaBasic Titles by Bryan Catley
A Public Domain Modula-2 System reviewed by Warren Block
One Drive Compile by Douglas Lovell
Using Lattice C with a single drive system
A Megabyte Without Megabucks by Chris Irving
An Internal Megabyte upgrade
Digi-View reviewed by Ed Jakober
Defender of the Crown reviewed by Keith Conforti
Leader Board reviewed by Chuck Raudonis
Roundhill Computer System's PANEL reviewed by Ray Lance
Digi-Paint.....by New Tek previewed by John Foust
Deluxe Paint II ....from Electronic Arts previewed by J. Foust



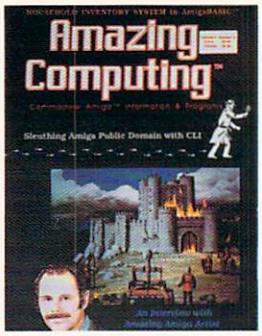
Volume 2 Number 2 1987

The Modem by Josph L. Rothman efforts of a BBS Sysop
MacroModem reviewed by Stephen R. Pietrowicz
GEMINI or "It takes two to Tango" by Jim Meadows
Gaming between machines
BBS-PC! reviewed by Stephen R. Pietrowicz
The Trouble with Xmodem by Joseph L. Rothman
The ACO Project....Graphic Teleconferencing on the Amiga
by S. R. Pietrowicz
Flight Simulator II....A Cros Country Tutorial by John Rafferty
A Disk Librarian in AmigaBASIC by John Kennan
Creating and Using Amiga Workbench Icons by C. Hansel
AmigaDOS version 1.2 by Clifford Kent
The Amazing MIDI Interface build your own by Richard Rae
AmigaDOS Operating System Calls and
Disk File Management by D. Haynie
Working with the Workbench by Louis A. Mamakos Prog in C



Volume 2 Number 3

The Amiga 2000™ by J Foust
A First look at the new, high end Amiga™
The Amiga 500™ by John Foust
A look at the new, low priced Amiga
An Analysis of the New Amiga PCs by J. Foust
Speculation on the New Amigas
Gemini Part II by Jim Meadows
The concluding article on two-player games
Subscripts and Superscripts in AmigaBASIC by Ivan C. Smith
The Winter Consumer Electronics Show by John Foust
AmigaTrix by W. Block Amiga™ shortcuts
Intuition Gadgets by Harriet Maybeck Tolly
A journey through gadget-land, using C
Shanghai reviewed by Keith M. Conforti
Chessmaster 2000 & Chessmate reviewedby Edwin V. Apel, Jr.
Zing! from Meridian Software reviewed by Ed Bercovitz
Forth! by Jon Bryan Get stereo sound into your Forth programs.
Assembly Language on the Amiga™ by Chris Martin
Roomers by theBandito Genlocks are finally shipping, & MORE!!!
AmigaNotes by R. Rae Hum Busters... "No stereo? Y not?...
The AMICUS Network by J. Foust
CES, user group issues and Amiga Expo"



Volume 2 Number 4 1987

Amazing Interviews Jim Sachs by S. Hull Amiga Artist
The Mouse That Got Restored by Jerry Hull and Bob Rhode
Sleuthing Public Domain Disks with CLI by John Foust
Highlights from the San Francisco Commodore Show
by S Hul
Speaker Sessions: San Francisco Commodore Show H Tolly
The Household Inventory System in AmigaBASIC™
by B Catley
Secrets of Screen Dumps by Natkun Okun
Using Function Keys with MicroEmacs by Greg Douglas
Amigatrix II by Warren Block More Amiga shortcuts
Basic Gadgets by Brian Catley Create gadget functions
Gridiron reviewed by K. Conforti Real football for the Amiga
Star Fleet I Version 2.1 reviewed by J. Tracy Amigain Space
The TIC reviewed by J. Foust Battery powered Clock Calendar
Metascope review by H. Tolly An easy-to-use debugger

Volume 2 Number 5 1987

The Perfect Sound Digitizer review by R. Battle
The Future Sound Digitizer by W. Block Applied Vision's SD
Forth! by J. Bryancomparing JForth amd Multi-Forth.
Basic Input by B. Catley AmigaBASIC input routine for use in
all your programs.
Writing a SoundScape Module in C by T. Fay Programming
with MIDI, Amiga and SoundScape by SoundScape author.
Programming in 68000 Assembly Language by C. Martin
Continuing with Counters & Addressing Modes.
Using FutureSound with AmigaBASIC by J. Meadows
AmigaBASIC Programming utility with real, digitized STEREO
AmigaNotes by R. Rae A review of Mimetics
SoundScape Sound Sampler.
More AmigaNotes by R. Rae
A further review of Sunrize's Perfect Sound .
Waveform Workshop in AmigaBASIC by J. Shields edit & save
waveform for use in other AmigaBASIC programs.
The Mimetics Pro MIDI Studio by Sullivan, Jeffery
A review of Mimetics' music editor/player.
Intuition Gadgets Part II by H. MaybeckTolly Boolean gadgets
provide the user with an on/off user interface.

Volume 2 Number 6 1987

Forth! by J. Bryan Access resources in the ROM Kernal.
The Amazing Computing Hard Disk Review by J. Foust & S.
Leemon In-depth looks at the C Ltd. Hard Drive, Microbotics'
MAS-Drive20, Byte by Byte's PAL Jr., Supra's 4x4 Hard Drive
and Xebec's 9720H Hard Drive. Also, a look at disk driver
software currently under devlopment.

Modula-2 AmigaDOS™ Utilities by S. Faiwiszewsk A
Calls to AmigaDOS and the ROM kernal.
Amiga Expansion Peripheral by J. Foust
Explanation of Amiga expansion peripherals.
Amiga Technical Support by J. Foust
How and where to get Amiga tech support.
Goodbye Los Gatos by J. Foust Closing Los Gatos.
The Amicus Network by J. Foust West Coast Computer Faire.
Metacomco Shell and Toolkit by J. Foust A review
The Magic Sac by J. Foust Run Mac programs on your Amiga.
What You Should Know Before Choosing an Amiga 1000
Expansion Device by S. Grant
7 Assemblers for the Amiga by G. Hull Choose your assembler
High Level Shakeup Replaces Top Management at
Commodore by S. Hull
Peter J. Baczor"by S. Hull Manager at CBM gives an inside look
Logistix A review by Richard Knepper
Organize! by A review Richard Knepper database.
68000 Assembly Language Programming on the Amiga
by Chris Martin
Superbase Personal Relational Database by Ray McCabe
AmigaNotes by Rae, Richard A look at FutureSound
Commodore Shows the Amiga 2000 and 500 at the Boston
Computer Society by H Maybeck Tolly

Volume 2, Number 7 1987

New Breed of Video Products by John Foust...
Very Vivid! by Tim Grantham...
Video and Your Amiga by Oran Sands III
Amigas & Weather Forecasting by Brenden Larson
A-Squared and the Live! Video Digitizer by John Foust.
Aegis Animator Scripts and Cel Animation by John Foust
Quality Video from a Quality Computer by Oran Sands III.
Is IFF Really a Standard? by John Foust..
Amazing Stories and the Amiga™ by John Foust.
All about Printer Drivers by Richard Bielak
Intuition Gadgets by Harriet Maybeck Tolley.
Deluxe Video 1.2 by Bob Eller
Pro Video CG1 by Oran Sands III.
Digi-View 2.0 Digitizer/Software by Jennifer M. Janik
Prism HAM Editor from Impulse by Jennifer M. Janik
Easyl drawing tablet by John Foust..
CSA's Turbo-Amiga Tower by Alfred Aburto
68000 Assembly Language by Chris Martin.

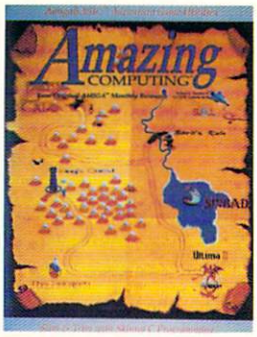
Your Resource to the Commodore Amiga™

The above phrase is much more than empty words. The pages of Amazing Computing™ are filled with articles on technical operations and procedures, basic use, and just-plain-fun. The growing library of Amazing Back Issues contains articles ranging from building your own IBM Disk controller, to setting up your own startup sequence. Amazing Computing™ has repeatedly been the first magazine to offer users solid, in-depth reviews and hands-on articles for their Amigas.

Amazing Computing™ was the first magazine to document CLI.
Amazing Computing™ was the first to show Sidecar™ from COMDEX™ in full detail.
Amazing Computing™ was the first to document a 5 1/4 drive connector.
Amazing Computing™ was the first with a 1 Meg Amiga upgrade hardware project!
Amazing Computing™ was the first magazine to offer serious programming examples and help.
Amazing Computing™ was the first magazine to offer Public Domain Software at reasonable prices.
Amazing Computing™ was the first magazine with the user in mind!

From the Beginning

Since February 1986, Amazing Computing™ has been providing users with complete information for their Amigas. This vast collection of programs and information is still available through our back issues. From the Premiere issue to the present, AC has been packed with Amiga insights that any user will find informative.

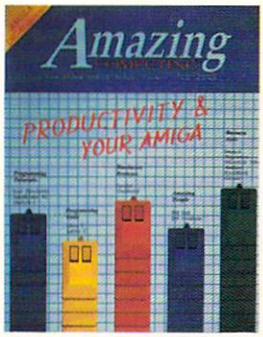


Volume 2, Number 8 1987

This month Amazing Computing™ focuses on entertainment packages for the Amiga. Amazing game reviews...

SDI, Earl Weaver Baseball, Portal, The Surgeon, Little Computer People, Sinbad, StarGlider, King's Quest I,II and III, Faery Tale Adventure, Ultima III, Facets of Adventure, Video Vegas and Bard's Tale.

Plus Amazing monthly columns... Amiga Notes, Roomers, Modula-2, 68000 Assembly Language and The Amicus Network.
Disk-2-Disk by Matthew Leeds
The ColorFonts Standard by John Foust
Skinny C Programs by Robert Riemersma, Jr.
Hidden Messages In Your Amiga™ by John Foust
The Consumer Electronics Show and Comdex by J Foust



Volume 2 Number 9 1987

Analyze 2.0 reviewed by Kim Schaffer
Impact Business Graphics review by Chuck Raudonis
Microfiche Filer review by Harv Laser
Pagesetter review by Rick Winch
Gizmoz Productivity Set 2.0 review by Bob Eller
Kickwork review by Harv Laser
Diga Telecommunications Package review by Steve Hull
Mouse Time and Timesaver review by John Foust
Insider Memory Expansion review by James O'Keane
Microbotics Starboard-2 review by S. Fawiszewski
Leather Goddesses of Phobos
reviewed by Harriet Maybeck-Tolly
Lattice C Compiler Version 3.10 reviewed by Gary Sarff
Manx 3.4a Update reviewed by John Foust
AC-BASIC reviewed by Sheldon Leemon
AC-BASIC Compiler an alternative comparison by B Catley
Modula-2 Programming S Fawiszewski
Raw Console Device Events
Directory Listings Under AmigaDOS by Dave Haynie
AmigaBASIC Patterns by Brian Catley
Programming with Soundscape
Todor Fay manipulates samples
Bill Volk, Vice-President Aegis Development,
interviewed by Steve Hull
Jim Goodnow, Developer of Manx 'C'
interview by Harriet M Tolly

To Be Continued.....

$4.00 each!

Our Back Issue price is still $4.00 per issue!

(Foreign orders, please add $1.00 per issue for Postage & Handling. All payments must be made by check or money order in U.S. funds drawn on a U.S. Bank.)

Limited Supply

Unfortunately, nothing lasts forever. The availabilty of some of our Back Issues is definitely limited. Please complete your Amazing Computing™ library today, while these issues are still available. Please complete the order form in the rear of this issue and mail with check or money order to:

Back Issues
PiM Publications, Inc.
P.O. Box 869
Fall River, MA 02722

(Please allow 4 to 6 weeks for delivery)

Amaze Me!

Please use this order form when subscribing to Amazing Computing™, ordering Back Issues, or ordering Amiga™ Public Domain Software

Name _____

Street _____

City _____ **St.** ____ **ZIP**_____

Amount enclosed _____

Please Check below.

☐ Please start my subscription to Amazing Computing™ with the next available issue. I have enclosed $24.00 for 12 issues in the U.S. ($30.00 Canada and Mexico, $35.00 overseas). All funds must be in U.S. Currency drawn on a U.S. Bank.

BACK ISSUES: $4.00 each
(foreign orders add $1.00 each for Postage and Handling)

Please circle your choices below.

VOL.1#1 VOL.1#2 VOL.1#3 VOL.1#4 VOL.1#5 VOL.1#6 VOL.1#7 VOL.1#8 VOL.1#9
VOL.2#1 VOL.2#2 VOL.2#3 VOL.2#4 VOL.2#5 VOL.2#6 VOL.2#7 VOL.2#8 VOL.2#9

Public Domain Software:

$6.00 each for subscribers (yes, even new ones!)
$7.00 each for non subscribers.

Please circle your choices below.

AMICUS:

A1 A2 A3 A4 A5 A6 A7 A8 A9 A10
A11 A12 A13 A14 A15 A16 A17 A18 A19 A20
A21 A22

Fred Fish:

FF1	FF2	FF3	FF4	FF5	FF6	FF7	FF8	FF9	FF10
FF11	FF12	FF13	FF14	FF15	FF16	FF17	FF18	FF19	FF20
FF21	FF22	FF23	FF24	FF25	FF26	FF27	FF28	FF29	FF30
FF31	FF32	FF33	FF34	FF35	FF36	FF37	FF38	FF39	FF40
FF41	FF42	FF43	FF44	FF45	FF46	FF47	FF48	FF49	FF50
FF51	FF52	FF53	FF54	FF55	FF56	FF57	FF58	FF59	FF60
FF61	FF62	FF63	FF64	FF65	FF66	FF67	FF68	FF69	FF70
FF71	FF72	FF73	FF74	FF75	FF76	FF77	FF78	FF79	FF80
FF81	FF82	FF83	FF84	FF85	FF86	FF87	FF88	FF89	FF90

The AMICUS & Fred Fish

Public Domain Software Library

This software is collected from user groups and electronic bulletin boards around the nation. Each Amicus disk is nearly full, and is fully accessible from the Workbench. If source code is provided for any program, then the executable version is also present. This means that you don't need the C compiler to run these programs. An exception is granted for those programs only of use to people who own a C compiler.

The Fred Fish disk are collected by Mr. Fred Fish, a good and active friend of the Amiga.

Note: Each description line below may include something like 'S-O-E-D', which stands for 'source, object file, executable and documentation'. Any combination of these letters indicates what forms of the program are present. Basic programs are presented entirely in source code format.

AMICUS Disk 1

ABasic programs: Graphics

3DSolids	3d solids modeling prog. w/sample data files
Blocks	draws blocks
Cubes	draws cubes
Durer	draws pictures in the style of Durer
FScape	draws fractal landscapes
Hidden	3D drawing program, w/ hidden line removal
JPad	simple paint program
Optical	draw several optical illusions
PaintBox	simple paint program
Shuttle	draws the Shuttle in 3d wireframe
SpaceArt	graphics demo
Speaker	speech utility
Sphere	draws spheres
Spiral	draws color spirals
ThreeDee	3d function plots
Topography	artificial topography
Wheels	draws circle graphics
Xenos	draws fractal planet landscapes

ABasic programs: Tools

AddressBook	simple database program for addresses
CardFile	simple card file database program
Demo	multiwindow demo
KeyCodes	shows keycodes for a key you press
Menu	run many ABasic programs from a menu
MoreColors	way to get more colors on the screen at once, using aliasing
shapes	simple color shape designer Speakit speech and narrator demo

ABasic programs: Games

BrickOut	classic computer brick wall game
Othello	also known as 'go'
Saucer	simple shoot-em-up game
Spelling	simple talking spelling game
ToyBox	selectable graphics demo

ABasic programs: Sounds

Entertainer	plays that tune
HAL9000	pretends it's a real computer
Police	simple police siren sound
SugarPlum	plays "The Dance of the Sugarplum Fairies"

C programs:

ATerm	simple terminal program, S-E
cc	aid to compiling with Lattice C
decvnt	opposite of CONVERT for cross developers
Dotty	source code to the 'dotty' window demo
echox	unix-style filename expansion, partial S,O-D
fastertp	explains use of fast-floating point math
FixDate	fixes future dates on all files on a disk, S-E
freedraw	simple Workbench drawing prog.,S-E
GfxMem	graphic memory usage indicator, S-E
Grep	searches for a given string in a file with
ham	shows off the hold-and-modify method of color generation
IBM2Amiga	fast parallel cable transfers between an IBM and an Amiga
Mandel	Mandelbrot set program, S-E
moire	patterned graphic demo, S-E
objfix	makes Lattice C object file symbols visible to Wack, S-E
quick	quick sort strings routine
raw	example sample window I/O
setiace	turns on interlace mode, S-E
sparks	qix-type graphic demo, S-E

Other executable programs:

SpeechToy	speech demonstration
WhichFont	displays all available fonts

Texts:

68020	describes 68020 speedup board from CSA
Aliases	explains uses of the ASSIGN command
Bugs	known bug list in Lattice C 3.02
CLICard	reference card for AmigaDOS CLI
CLICommands	guide to using the CLI
Commands	shorter guide to AmigaDOS CLI commands
EdCommands	guide to the ED editor
Filenames	AmigaDOS filename wildcard conventions
HalfBright	explains rare graphics chips that can do more colors
ModemPins	description of the serial port pinout
RAMdisks	tips on setting up your RAM: disk
ROMWack	tips on using ROMWack
Sounds	explanation of Instrument demo sound file format
Speed	refutation of Amiga's CPU and custom chip speed
WackCmds	tips on using Wack

AMICUS Disk 2

C programs:

alib	AmigaDOS object library manager , S-E
ar	text file archive program, S-E
fixobj	auto-chops executable files
shell	simple CLI shell, S-E
sq, usq	file compression programs, S-E
YachtC	a familiar game, S-E
Make	a simple 'make' programming utility, S-E
Emacs	an early version of the Amiga text editor, S-E-D

Assembler programs:

bsearch.asm	binary search code
qsort.asm	Unix compatible qsort() function, source and C test program
setjmp.asm	setjmp() code for Lattice 3.02
SVprintf	Unix system V compatible printf()
trees.o	Unix compatible tree() function, O-D

(This disk formerly had IFF specification files and examples. Since this spec is constantly updated, the IFF spec files have been moved to their own disk in the AMICUS collection.)

John Draper Amiga Tutorials:

Animate	describes animation algorithms
Gadgets	tutorial on gadgets
Menus	learn about Intuition menus

AMICUS Disk 3

C programs:

Xref	a C cross-reference gen., S-E
6bitcolor	extra-half-bright chip gfx demo, S-E
Chop	truncate (chop) files down to size, S-E
Cleanup	removes strange characters from text files
CR2LF	converts carriage returns to line feeds in Amiga files, S-E
Error	adds compile errors to a C file, S
Hello	window ex. from the RKM, S
Kermit	generic Kermit implementation, flakey, no terminal mode, S-E
Scales	sound demo plays scales, S-E
SkewB	Rubik cube demo in hi-res colors, S-E

AmigaBasicProgs(dir)

Automata	cellular automata simulation
CrazyEights	card game
Graph	function graphing programs
WitchingHour	a game

ABasiC programs:

Casino	games of poker, blackjack, dice, and craps
Gomoku	also known as 'othello'
Sabotage	sort of an adventure game

Executable programs:

Disassem	a 68000 disassembler, E-D
DpSlide	shows a given set of IFF pictures, E-D
Arrange	a text formatting program, E-D

Assembler programs:

Argoterm	terminal program with speech and Xmodem, S-E

AMICUS Disk 4 Files from the original Amiga Technical BBS

Note that some of these files are old, and refer to older versions of the operating system. These files came from the Sun system that served as Amiga technical support HQ for most of 1985. These files do not carry a warranty, and are for educational purposes only. Of course, that's not to say they don't work.

Complete and nearly up-to-date C source to 'image.ed', an early version of the Icon Editor. This is a little flaky, but compiles and runs.

An Intuition demo, in full C source, including files: demomenu.c, demomenu2.c, demoreq.c, getascii.c, idemo.c, idemo.guide, idemo.make, idemoall.h, nodos.c, and txwrite.c

addmem.c	add external memory to the system
bobtest.c	example of BOB use
consoleIO.c	console IO example
creaport.c	create and delete ports
creastdi.c	create standard I/O requests
creatask.c	creating task examples
diskio.c	example of track read and write
dotty.c	source to the 'dotty window' demo
dualplay.c	dual playfield example
flood.c	flood fill example
freemap.c	old version of 'freemap'
getbobs.c	tools for VSprites and BOBs
gfxmem.c	graphic memory usage indicator
hello.c	window example from RKM
inputdev.c	adding an input handler to the input stream
joystik.c	reading the joystick
keybd.c	direct keyboard reading
layertes.c	layers examples
mousport.c	test mouse port
ownlib.c, ownlib.asm	example of making your own library with Lattice
paratest.c	tests parallel port commands
sertest.c	tests serial port commands
serisamp.c	example of serial port use
prinintr.c	sample printer interface code
prtbase.h	printer device definitions
regintes.c	region test program
setiace.c	source to interlace on/off program
setparallel.c	set the attributes of the parallel port
SetSerial.c	set the attributes (parity, data bits, baud rate, etc.) of the serial port
singplay.c	single playfield example
speechtoy.c	source to narrator and phonetics demo
timedely.c	simple timer demo
timer.c	exec support timer functions
timrstuf.c	more exec support timer functions
WhichFont.c	loads and displays all available system fonts
process.i and prtbase.i	assmebler include files:
autorqstr.txt	warnings of deadlocks with autorequesters
consoleIO.txt	copy of the RKM console I/O chapter
diskfont.txt	warning of disk font loading bug
fullfunc.txt	list of #defines, macros, functions
inputdev.txt	preliminary copy of the input device chapter
License	information on Workbench distribution license
printer	pre-release copy of the chapter on printer drivers, from RKM 1.1
v11fd.txt	'diff' of .fd file changes from version 1.0 to 1.1
v28v1.diff	'diff' of include file changes from version 28 to 1.0

AMICUS Disk 5 Files from the Amiga Link / Amiga Information Network

Note that some of these files are old, and refer to older versions of the operating system. These files are from Amiga Link. For a time, Commodore supported Amiga Link, aka AIN, for online developer technical support. It was only up and running for several weeks. These files do not carry a warranty, and are for educational purposes only. Of course, that's not to say they don't work.

A demo of Intuition menus called 'menudemo', in C source

whereis.c	find a file searching all subdirectories
bobtest.c	BOB programming example
sweep.c	sound synthesis example

Assembler files:

mydev.asm	sample device driver
mylib.asm	sample library example
mylib.i	
mydev.i	
asmsupp.i	
macros.i	assembler include files

Texts:

amigatricks	tips on CLI commands
extdisk	external disk specification
gameport	game port spec
parallel	parallel port spec
serial	serial port spec
v1.1update	list of new features in version 1.1
v1.1h.txt	'diff' of include file changes from v1.0 to v1.1h version

Files for building your own printer drivers, including dospecial.c, epsondata.c, init.asm, printer.c, printer.link, printertag.asm, render.c, and wait.asm. This disk does contain a number of files describing the IFF specification. These are not the latest and greatest files, but remain here for historical purposes. They include text files and C source examples. The latest IFF spec is elsewhere in this library.

AMICUS Disk 6 IFF Pictures

This disk includes the DPSlide program, which can view a given series of IFF pictures, and the 'showpic' program, which can view each file at the click of an icon. The pictures include a screen from ArticFox, a Degas dancer, the guys at Electronic Arts, a gorilla, horses, King Tut, a lighthouse, a screen from Marble Madness, the Bugs Bunny Martian, a still from an old movie, the Dire Straits moving company, a screen from Pinball Contruction Set, a TV newcaster, the PaintCan, a world map, a Porsche, a shuttle mission patch, a tyrannosaurus rex, a planet view, a VISA card, and a ten-speed.

AMICUS Disk 7 DigiView HAM demo picture disk

This disk has pictures from the DigiView hold-and-modify video digitizer. It includes the ladies with pencils and lollypops, the young girl, the bulldozer, the horse and buggy, the Byte cover, the dictionary page, the robot and Robert. This includes a program to view each picture separately, and all together as separate, slidable screens. The 'seeibm' program, to turn any screen into an IFF picture.

AMICUS Disk 8

C programs:

Browse	view text files on a disk, using menus S-E-D
Crunch	removes comments and white space from C files, S-E
IconExec	EXECUTE a series of commands from Workbench S-E
PDScreen Dump	dumps Rastport of highest screen to printer
SetAltemate	sets a second image for an icon, when clicked once S-E
SetWindow	makes windows for a CLI program to run under Workbench S-E
SmallClock	a small digital clockin a window menu bar
Scrimper	the screen printer in the fourth AC S-E

Amiga Basic Programs:

(Note: Many of these programs are present on AMICUS Disk 1. Several of these were converted to Amiga Basic, and are included here.)

AddressBook	a simple address book database
Ball	draws a ball
Cload	program to convert Compuserve hex files to binary, S-D
Clue	the game, Intuition driven
ColorArt	art drawing program
DeluxeDraw	the drawing program in the 3rd AC, S-D
Eliza	conversational computer psychologist
Othello	the game, as known as 'go'
RatMaze	3D ratmaze game
ROR	boggling graphics demo
Shuttle	draws 3D pictures of the space shuttle
Spelling	simple spelling program
YoYo	wierd zero-gravity yo-yo demo, tracks yo-yo to the mouse

Executable programs:

3Dcube	Modula-2 demo of a rotating cube
AltIcon	sets a second icon image, displayed when the icon is clicked
AmigaSpell	a slow but simple spell checker, E-D
arc	the ARC file compression progra must-have for telecom, E-D
Bertrand	graphics demo
disksalvage	prog. to rescue trashed disks, E-D
KwikCopy	a quick but nasty disk copy program: ignores errors, E-D
LibDir	lists hunks in an object file E-D
SaveILBM	saves any screen as IFF pic.E-D ??
ScreenDump	shareware screen dump prog, E only
StarTerm	version 2.0, term program, XmodemE-D

Texts:

LatticeMain	tips on fixing _main.c in Lattice
GDiskDrive	make your own 5 1/4 drive
GuruMed	explains the Guru numbers
Lat3.03bugs	bug list of Lattice C version 3.03
MForgeRev	user's view of the MicroForge HD
PrintSpooler	EXECUTE-based print spool prog.

.BMAP files:

These are the necessary links between Amiga Basic and the system libraries. To take advantage of the Amiga's capabilites in Basic, you need these files. BMAPs are included for 'dist', 'console', 'diskfont', 'exec', 'icon', 'intuition', 'layers', 'mathffp', mathieeedoubas', 'mathieeesingbas', 'mathtrans', 'potgo', 'timer' and 'translator'.

AMICUS Disk 9

Amiga Basic Programs:

FlightSim	simple flight simulator program
HuePalette	explains Hue, Saturation, & Intensity
Requester	ex. of requesters from Amiga Basic
ScrollDemo	demonstrates scrolling capabilites
Synthesizer	sound program
WorldMap	draws a map of the world

Executable programs:

Boing!	latest Boing! demo,with selectable speed,E
Brush2C	converts an IFF brush to C data instructions, initialization code, E
Brush2Icon	converts IFF brush to an icon, E
Dazzle	graphics demo, tracks to mouse, E
DeciGEL	assembler program for stopping 68010 errors, S-E-D
Klock	menu-bar clock and date display, E
life	the game of life, E
TimeSet	Intuition-based way to set the time & date
EMEmacs	another Emacs, more oriented to word processing, S-E-D
MyCLI	a CLI shell, works without the Workbench, S-E-D

Texts:

FnctnKeys	read function keys from Amiga Basic
HackerSin	explains how to win the game 'hacker'
Ist68010	guide to installing a 68010 in your Amiga
Boing!	latest Boing! demo,with selectablespeed, E
Brush2C	converts an IFF brush to C data instructions, initialization code, E
Brush2Icon	converts IFF brush to an icon, E
Dazzle	graphics demo, tracks to mouse, E
DeciGEL	assembler program for stopping 68010 errors, S-E-D
Klock	menu-bar clock and date display, E
life	the game of life, E
TimeSet	Intuition-based way to set the time date,
EMEmacs	another Emacs, more oriented to word processing, S-E-D
MyCLI	a CLI shell, works without the Workbench, S-E-D

Texts:

FnctnKeys	explains how to read function keys from Amiga Basic
HackerSin	explains how to win the game 'hacker'
list68010	guide to installing a 68010 in your Amiga
PrinterTip	sending escape sequences to your printer
StartupTip	tips on setting up your startup-sequence file
XfrmrReview	list of Transformer programs that work

Printer Drivers:

Printer drivers for the Canon PJ-1080A, the C Itoh Prowriter, an improved Epson driver that eliminates streaking, the Epson LQ-800, the Gemini Star-10, the NEC 8025A, the Okidata ML-92, the Panasonic KX-P10xx family, and the Smith-Corona D300, with a document describing the installation process.

AMICUS Disk 10 Instrument sound demos

This is an icon-driven demo, circulated to many dealers. It includes the sounds of an acoustic guitar, an alarm, a banjo, a bass guitar, a boink, a calliope, a car horn, claves, water drip, electric guitar, a flute, a harp arpegio, a kickdrum, a marimba, a organ minor chord, people talking, pigs, a pipe organ, a Rhodes piano, a saxophone, a sitar, a snare drum, a steel drum, bells, a vibrophone, a violin, a wailing guitar, a horse whinny, and a whistle.

AMICUS Disk 11

C programs

dirutil	Intuition-based, CLI replacement manager S-E
cpri	shows and adjusts priority of CLI processes, S-E
ps	shows info on CLI processes, S-E
vidtex	displays Compuserve RLE pics, S-E

AmigaBasic programs

pointered	pointer and sprite editor program
optimize	optimization ex ample from AC article
calendar	large, animated calendar, diary and date book program
amortize	loan amortizations
brushtoBOB	converts small IFF brushes to AmigaBasic BOB OBJECTS
grids	draw and play waveforms
hilbert	draws Hilbert curves
madlib	mad lib story generator
mailtalk	talking mailing list program
meadows3D	3D graphics program, from A C™ article
mousetrack	mouse tracking example in hires mode
slot	slot machine game
tictactoe	the game
switch	pachinko-like game
weird	makes strange sounds

Executable programs

cp	unix-like copy command, E
cls	screen clear, S-E
diff	unix-like stream editor uses 'diff' output to fix files
pm	chart recorder performances indicator

Assembler programs

cls	screen clear and CLI arguments example

Modula-2

trails	moving-worm graphics demo
caseconvert	converts Modula-2 keywords to uppercase

Forth

Breshehan	circle algorithm example
Analyze	12 templates for the spreadsheet Analyze

There are four programs here that read Commodore 64 picture files. They can translate Koala Pad, Doodle, Print Shop and News Room graphics to IFF format. Getting the files from your C-64 to your Amiga is the hard part.

AMICUS Disk 12

Executable programs

blink	'alink' compatible linker, but faster, E-D
clean	spins the disk for disk cleaners, E-D
epsonset	sends Epson settings to PAR from menu E-D
showbig	view hi-res pics in low-res superbitmap, E-D
speaktime	tell the time, E-D
undelete	undeletes a file, E-D
cnvapidhm	converts Apple][low, medium and high res pictures to IFF, E-D
menued	menu editor produces C code for menus, E-D
quick	quick disk-to-disk nibble copier, E-D
quickEA	copies Electronic Arts disks, removes protection, E-D
bed 1.3	demo of text editor from Microsmiths,E-D

C programs

spin3	rotating blocks graphics demo, S-E-D
popdi	start a new CLI at the press of a button, like Sidekick, S-E-D
vsprite	VSprite example code from Commodore, S-E-D
AmigaBBS	Amiga Basic bulletin board prog., S-D

Assembler programs

star10	makes star fields like Star Trek intro,S-E-D

Pictures

Mount Mandelbrot	3D view of Mandelbrot set
Star Destroyer	hi-res Star Wars starship
Robot	robot arm grabbing a cylinder

Texts

vendors	Amiga vendors, names, addresses
cardco	fixes to early Cardco memory boards
cinclude	cross-reference to C include files
mindwalker	clues to playing the game well
slideshow	make your own slideshows from the Kaleidoscope disk

AMICUS Disk 13

Amiga Basic programs

Routines from Carolyn Schepper of CBM Tech Support, to read and display IFF pictures from Amiga Basic. With documentation. Also included is a program to do screen prints in Amiga Basic, and the newest BMAP files, with a corrected ConvertFD program. With example pictures, and the SaveILBM screen capture program.

Routines to load and play FutureSound and IFF sound files from Amiga Basic, by John Foust for Applied Visions. With documentation and C and assembler source for writing your own libraries, and interfacing C to assembler in libraries. With example sound.

Executable programs

gravity	Sci Amer Jan 86 gravitation graphic simulation, S-E-D

Texts

MIDI	make your own MIDI instrument interface, with documentation and a hi-res schematic picture.

AMICUS Disk 14

Several programs from Amazing Computing issues:

Tools

Dan Kary's	C structure index program, S-E-D

Amiga Basic programs:

BMAP Reader	by Tim Jones
IFFBrush2BOB	by Mike Swinger
AutoRequester	example
DOSHelper	Windowed help system for CLI commands, S-E-D
PETrans	translates PET ASCII files to ASCII files, S-E-D
C Squared	Graphics program from Scientific American, Sept 86, S-E-D
crlf	adds or removes carriage returns from files, S-E-D
dpdecode	decrypts Deluxe Paint, remo
ves copy	protection, E-D
queryWB	asks Yes or No from the user returns exit code, S-E
vc	VisiCalc type spreadsheet, no mouse control, E-D
view	views text files with window and
slider	gadget, E-D

Oing, Sproing, yaBoing, Zoing are sprite-based Boing! style demos, S-E-D

CLIClock, sClock, wClock are window border clocks, S-E-D

Texts

An article on long-persistance phospor monitors, tips on making brushes of odd shapes in Deluxe Paint, and recommendations on icon interfaces from Commodore-Amiga.

AMICUS 15

The C programs include:

'pr'	a file printing utility, which can print files in the background, and with line numbers and control character filtering.
'fm'	displays a chart of the blocks allocated on a disk.
'Ask'	questions an 'execute' file, returns an error code to control the execution in that batch file
'Stat'	an enhanced version of AmigaDOS 'status' command.
'Dissolve'	random-dot dissolve demo displays IFF picture slowly, dot by dot, in a random fashion.
'PopCLI2'	invoke new CLI window at the press of a key.

The executable programs include:

'Form'	file formatting program through the printer driver to select print styles
'DiskCat'	catalogs disks, maintains, sorts,merges lists of disk files
'PSound'	SunRize Industries' sampled sound editor & recorder
'Iconmaker'	makes icons for most programs
'Fractals'	draws great fractal seascapes and mountain scapes.
'3D Breakout'	3D glasses, create breakout in a new dimension
'AmigaMonitor'	displays lists of open files,
memory use,	tasks, devices and ports in use.
'Cosmoroids'	version of 'asteroids' for the Amiga.
'Sizzlers'	high resolution graphics demo written in Modula 2.

Texts:

'ansi.txt'	explains escape sequences the CON: device responds to.
'FKey'	includes template for making paper to sit in the tray at the top of the Amiga keyboard.
'Spawn'	programmer's document from Commodore Amiga, describe ways to use the Amiga's multitasking capabilities in your own programs.

AmigaBasic programs:

'Grids'	draw sound waveforms, and hear them played.
'Light'	a version of the Tron light-cycle video game.
'MigaSol'	a game of solitaire.
'Stats'	program to calculate batting averages
'Money'	"try to grab all the bags of money that you can."

AMICUS 15 also includes two beautiful IFF pictures, of the enemy walkers from the ice planet in Star Wars, and a picture of a cheetah.

AMICUS 16

'juggler'	demo by Eric Graham. a robot juggler bouncing three mirrored balls, with sound effects. Twenty-four frames of HAM animation are flipped quickly to produce this image. You control the speed of the juggling. The author's documentation hints that this program might someday be available as a product.

IFF pictures

	parodies of the covers of Amiga World and Amazing Computing magazines.

C programs:

'inputhandler'	example of making an input handler.
'FileZap3'	binary file editing program
'ShowPrint'	displays IFF picture, and prints it.
'Gen'	program indexes and retrieves C structures and variables declared in the Amiga include file system.

Executable Programs:

'FixHunk2'	repairs an executable program file for expanded memory
'ms2smus'	converts Music Studio files to IFF standard 'SMUS' format. I have heard this program might have a few bugs, especially in regards to very long songs, but it works in most cases.
'Missile'	Amiga version of the 'Missile Command' video game,

This disk also contains several files of scenarios for Amiga Flight Simulator II. By putting one of these seven files on a blank disk, and inserting it in the drive after performing a special command in this game, a number of interesting locations are preset into the Flight Simulator program. For example, one scenario places your plane on Alcatraz, while another puts you in Central Park

AMICUS 17

Telcommunications disk which contains six terminal programs.

"Comm" V1.33	term prog. with Xmodem, WXmodem,
"ATerm" V7.2	term prog. includes Super Kermit
"VT-100"V2.6	Dave Wecker's VT-100 emulator with Xmodem,Kermit, and scripting
"Amiga Kermit"	V4D(060) port of the Unix C-Kermit
"VTek" V2.3.1	Tektronix graphics terminal emulator based on the VT-100 prog. V2.3 and contains latest 'arc' file compression
"AmigaHost"	V0.9 for Compuserve. Includes RLE graphics abilities & CIS-B file transfer protocol.
"FixHunk"	expansion memory necessity
"FixOq"	removes garbage characters from modem received files
"Txt"	filters text files from other systems to be read by the Amiga E.C.
"addmem"	executeable version for use with mem expansion article in AC v2..1
'arc'	file documentation and a basic tutorial on un 'arcing files
"arcre"	for makeing "arc" files E.C.

AMICUS Disk 18

Logo	Amiga version of the popular computer language, with example programs, E-D
TV'Text	Demo version of the TV'Text character generator
PageSetter	Freely distributable versions of the updated PagePrint and PageIFF programs for the PageSetter desktop publishing package.
FullWindow	Resizes any CLI window using only CLI commands, E-D
Life3d	3-D version of Conway's LIFE program, E-D
Defdisk	CLI utility to re-assign a new Workbench disk, S-E-D
Calendar.WKS	Lotus-compatible worksheet that makes calendars
SetKey	Demo of keyboard key re-programmer, with IFF picture to make function key labels, E-D
VPG	Video pattern generator for aligning monitors, E-D
HP-10C	Hewlett-Packard-like calculator, E-D
SetPrefs	Change the Preferences settings on the fly, in C, S-E-D
StarProbe	Program studies stellar evolution. C source included for Amiga and MS-DOS, S-E-D
ROT	C version of Colin French's AmigaBasic ROT program from Amazing Computing. ROT edits and displays polygons to create three dimensional objects. Up to 24 frames of animation can be created and displayed. E-D
Scat	Like Ing, windows on screen run away from the mouse, E-D
DK	Decays" the CLI window into dust, in Modula 2, S-E-D
DropShadow2	Adds layered shadows to Workbench windows, E-D

AMICUS 19

This disk carries several programs from Amazing Computing. The IFF pictures on this disk include the Amiga Wake part T-shirt logo, a sixteen-color hi-res image of Andy Griffith, and five Amiga Livel pictures from the Amazing Stories episode that featured the Amiga.

Solve	Linear equation solver in assembly language, S-E-D
Gadgets	Bryan Catley's AmigaBasic tutorial,
Household	Bryan Catley's AmigaBasic household inventory program, S-D
Waveform	Jim Shields' Waveform WorkAmigaBasic, S-D
DiskLib	John Kennan's AmigaBasic disk librarian program, S-D
Subscripts	Ivan Smith's AmigaBasic subscript example, S-D
String, Boolean	C programs and executables for Harriet Mayback Tolly's Intuition tutorials, S-E-D
Skinny C	Bob Riemersma's example for making small C programs, S-E-D
COMAL.h	Make C look like COMAL HeaderFile,
EmacsKey	Makes Emacs function key definitions by Greg Douglas, S-D
AMon 1.1	Snoop on system resource use, E-D
BTE	Bard's Tale character editor, E-D
Size	CLI program shows the size of a given set of files, E-D
WinSize	CLI window utility resizes current window, S-E-D

Disk 20

Compector, Decoder	Steve Michel AmigaBasic tools, S-D
BobEd	BOB and sprite editor written in C,S-E-D
SpriteMasterII	Sprite editor and animator by Brad Kiefer, E-D
BitLab	Bitter chip exploration C program by Tomas Rokicki, S-E-D
FPic	Image processing program by Bob Bush loads and saves IFF images, changes them with several techniques, E-D
Bankn	Complete home banking program, balance your checkbook! E-D
cons	Console device demo program with supporting macro routines.
freemap	Creates a visual diagram of free memory
input.dev	sample input handler, traps key or mouse events
joystick	Shows how to set up the gameportdevice as a joystick.
keyboard	demonstrates direct communications with the keyboard.
layers	Shows use of the layers library
mandelbrot	IFF Mandelbrot program
mouse	hooks up mouse to right joystick port
one.window	console window demo
parallel	Demonstrates access to the parallel port.
printer	opening and using the printer, does a screen dump, not working
print.support	Printer support routines, not working.
prodtest	sample process creation code, not working
region	demos split drawing regions
samplefont	sample font with info on creating your own
serial	Demos the serial port
singlePlayfield	Creates 320 x 200 playfield
speechtoy	latest version of cute speech demo
speech.demo	simplified version of speechtoy, with ID requests
text.demo	displays available fonts
timer	demos timer.device use
trackdisk	demos trakcdisk driver

AMICUS 21

Target	Makes each mouse click sound like a gunshot, S-E-D
Sand	Simple game of sand that follows the mouse pointer, E-D
PropGadget	Harriet Mayback Tolly's proportional gadget example, S-E
EHB	Checks to see if you have extra-half-bright graphics, S-E-D
Piano	Simple piano sound program
CelScripts	Makes cel animation scripts for Aegis Animator, in AmigaBasic

This disk has electronic catalogs for AMICUS disks 1 to 20 and Fish disks 1 to 80. They are viewed with the DiskCat program, included here.

AMICUS 22

Cycles	Light cycle game, E-D
Show_Printll	Views and prints IFF pictures, including larger than screen
PrtDrvGen2.3	Latest version of a printer driver generator
Animations	VideoScape animations of planes and boing ball
Garden	Makes fractal gardenscapes
BasicSorts	Examples of binary search and insertion sort in AmigaBasic

Fred Fish Public Domain Software

Fred Fish Disk 1:

amigademo	Graphical benchmark for comparing amigas.
amigaterm	simple communications program with Xmodem
balls	simulation of the "kinetic thingy" with balls on strings
colorful	Shows off use of hold-and-modify mode.
dhrystone	Dhrystone benchmark program.
dotty	Source to the "dotty window" demo on the Workbench disk.
freedraw	A small "paint" type program with lines, boxes, etc.
gad	John Draper's Gadget tutorial program
gfxmem	Graphical memory usage display prog.
halfbrite	demonstrates "Extra-Half-Brite" mode, if you have it
hello	simple window demo
lattfp	accessing the Motorola Fast Floating Point library from C
palette	Sample prog. to design color palettes.
trackdisk	Demonstrates use of the trackdisk driver.
requesters	John Draper's requester tutorial and example program.
speech	Sample speech demo program. Stripped down "speechtoy".
speechtoy	Another speech demo program.

Fred Fish Disk 2:

alib	Object module librarian.
cc	Unix-like frontend for Lattice C compiler.
dbug	Macro based C debugging package. Machine independent.
make	Subset of Unix make command.
make2	Another make subset command.
microemacs	Small version of emacs editor, with macros, no extensions
portar	Portable file archiver.
xrf	DECUS C cross reference utility.

Fred Fish Disk 3:

gothic	Gothic font banner printer.
roff	A "roff" type text formatter.
ff	A very fast text formatter
cforth	A highly portable forth implementation. Lots of goodies.
xlisp	Xlisp 1.4, not working correctly.

Fred Fish Disk 4:

banner	Prints horizontal banner
bgrep	A Boyer-Moore grep-like utility
bison	CNU Unix replacement 'yacc', not working.
bm	Another Boyer-Moore grep-like utility
grep	DECUS grep
kermit	simple portable Kermit with no connect mode.
MyCLI	Replacement CLI for the Amiga. V. 1.0
mandel	A Mandelbrot set program, by Robert French and RJ Mical

Fred Fish Disk 5

cons	Console device demo program with supporting macro routines.
freemap	Creates a visual diagram of free memory

For PDS orders, Please use form on page 106

input.dev	sample input handler, traps key or mouse events
joystick	Shows how to set up the gameport device as a joystick.
keyboard	demonstrates direct communications with the keyboard.
layers	Shows use of the layers library
mandelbrot	IFF Mandelbrot program
mouse	hooks up mouse to right joystick port
one.window	console window demo
parallel	Demonstrates access to the parallel port.
printer	opening and using the printer, does a screen dump, not working
print.support	Printer support routines, not working.
proctest	sample process creation code, not working
region	demos split drawing regions
samplefont	sample font with info on creating your own
serial	Demos the serial port
singlePlayfield	Creates 320 x 200 playfield
speechtoy	latest version of cute speech demo
speech.demo	simplified version of speechtoy, with IO requests
text.demo	displays available fonts
timer	demos timer.device use
trackdisk	demos trakcdisk driver

Fred Fish Disk 6:

compress	like Unix compress, a file squeezer
dadc	analog clock impersonator
microemacs	upgraded version of microemacs from disk 2
mult	removes multiple occuring lines in files
scales	demos using sound and audio functions
setparallel	Allows changing parallel port parameters
setserial	Allows changing serial port parameters.
sortc	quicksort based sort program, in C
stripc	Strips comments and extra whitespace from C source

Fred Fish Disk 7:

This disk contains the executables of the game Hack V 1.0.1.

Fred Fish Disk 8:

This disk contains the C source to Hack on disk 7.

Fred Fish Disk 9:

moire	Draws moire patterns in black and white
MVP-FORTH	Mountain View Press Forth, version 1.00.03A. A shareware version of FORTH from Fantasia Systems.
proff	a more powerful text formatting program
setiace	Program to toggle interlace mode on and off.
skewb	a rubic's cube type demo
sparks	moving snake Graphics demo

Fred Fish Disk 10:

conquest	An interstellar adventure simulation game
dehex	convert a hex file to binary
filezap	Patch program for any type of file.
fixobj	Strip garbage off Xmodem transferred files.
iff	Routines to read and write iff format files.
ld	simple directory program
ls	Minimal UNIX ls, with Unix-style wildcarding, in C
sq,usq	file squeeze and unsqueeze
trek73	Star Trek game
yachtc	Dice game.

Fred Fish Disk 11:

dpslide	slide show program for displaying IFF images with miscellaneous pictures

Fred Fish Disk 12:

amiga3d	Shows a rotating 3 dimensional solid "Amiga sign".
ArgoTerm	a terminal emulator program, written in assembler
arrow3d	Shows a rotating 3 dimensional wire frame arrow.
ld4	directory listing program
IconExec	
SetWindow	two progs for launching progs from Workbench, presently only works under CLI.
SetAlternate	Makes an icon show a second image when clicked once
StarTerm	terminal emulator, with ASCII Xmodem, dialer, more.

Fred Fish Disk 13:

A Bundle of Basic programs, including:

Jpad	toybox	ezspeak
mandlebrot	xmodem	3dsolids
addbook	algebra	ror
amgseq1	amiga-copy	band
bounce	box	brickout
canvas	card1	circle
colorcircles	Copy	cubes1
cutpaste	date	dogstar
dragon	draw	dynamictriangle
Eliza	ezterm	filibuster
fractal	fscape	gomoku
dart	haiku	hal9000
hailey	hauntedM	hidden
join	loz	mandel
menu	minipaint	mouse
Othello	patch	pena
pinwheel	gbox	random-circles
Readme	rgb	rgbtest
Rord	sabotage	sailestalk
shades	shapes	shuttle
sketchpad	spaceart	speakspeach
speecheasy	spell	sphere
spiral	striper	superped
suprshr	talk	terminal
termtest	tom	topography
triangle	wheels	xenos
xmostriper		

(note: some programs are Abasic, most are Amigabasic, and some programs are presented in both languages)

Fred Fish Disk 14:

amiga3d	update of #12, includes C source to a full hidden surface removal and 3D graphics
beep	Source for a function that generates a beep sound
dex	extracts text from within C source files
dimensions	demonstrates N dimensional graphics
filezap	update of disk 10, a file patch utility
gfxmem	update of disk 1, graphic memory usage indicator
gi	converts IFF brush files to Image struct, in Ctext.
pdterm	simple ANSI VT100 terminal emulator, in 80 x 25 screen
shell	simple Unix 'csh' style shell
termcap	mostly Unix compatible 'termcap' implementation.

Fred Fish Disk 15:

Blobs	graphics demo, like Unix 'worms'
Clock	simple digital clock program for the title bar
Dazzle	An eight-fold symmetry dazzler program. Really pretty!
Fish	double buffered sequence cycle animation of a fish
Monopoly	A really nice monopoly game written in AbasiC.
OkidataDump	Okidata ML92 driver and WorkBench screen dump program.
Polydraw	A drawing program written in AbasiC.
Polyfractals	A fractal program written in AbasiC.

Fred Fish Disk 16:

A complete copy of the latest developer IFF disk

Fred Fish Disk 17:

The NewTek Digi-View video digitizer HAM demo disk

Fred Fish Disk 18:

AmigaDisplay	dumb terminal program with bell, selectable fonts
Ash	Prerelease C Shell-like shell program, history, loops, etc.
Browser	wanders a file tree, displays files, all with the mouse
MC68010	docs on upgrading your Amiga to use a MC68010
Multdim	rotate an N dimensional cube with a joystick
PigLatin	SAY command that talks in Pig Latin
Scrimper	Screen image printer
Xlisp1.6	source, docs, and executable for a Lisp interpreter.

Fred Fish Disk 19:

BlackJack	text-oriented blackjack game
JayMinerSlides	Slides by Jay Miner, Amiga graphics chip designer, showing flowchart of the Amiga internals, in 640 x 400.
Keymap_Test	test program to test the key mapping routines
LockMon	Find unclosed file locks, for programs that don't clean up.

Fred Fish Disk 20:

AmigaToAtari	converts Amiga object code to Atari format
DiskSalv	program to recover files from a trashed AmigaDOS disk.
Hash	example of the AmigaDOS disk hashing function
Hd	Hex dump utility ala Computer Language magazine, April 86
MandelBrots	Mandelbrot contest winners
MultTasking	Tutorial and examples for Exec level multitasking
Pack	strips whitespace from C source
PortHandler	sample Port-Handler program that performs. Shows BCPL environment clues.
Random	Random number generator in assembly, for C or assembler.
SetMouse2	sets the mouse port to right or left
SpeechTerm	terminal Emulator with speech capabilities, XModem
TxEd	Demo editor from Microsmith's Charlie Heath

Fred Fish Disk 21

This is a copy of Thomas Wilcox's Mandelbrot Set Explorer disk. Very good!

Fred Fish Disk 22

This disk contains two new "strains" of microemacs.

Lemacs	version 3.6 by Daniel Lawrence. For Unix V7, BSD 4.2, Amiga, MS-DOS, VMS. Uses Amiga function keys, status line, execute, startup files, more.
Pemacs	By Andy Poggio. New features include <ALT> keys as Meta keys, mouse support, higher priority, backup files, word wrap, function keys.

Fred Fish Disk 23

Disk of source for MicroEmacs, several versions for most popular operating systems on micros and mainframes. For people who want to port MicroEmacs to their favorite machine.

Fred Fish Disk 24:

Conques	interstaller adventure simulation game
Csh	update to shell on Disk 14, with built in commands,named variables substitution.
Modula-2	A pre-release version of the single pass Modula-2 compiler originally developed for Macintosh at ETHZ. This code was transmitted to the AMIGA and is executed on the AMIGA using a special loader. Binary only.

Fred Fish Disk 25

Graphic Hack	A graphic version of the game on disks 7 and 8. This is the graphics-oriented Hack game by John Toebes. Only the executable is present.

Fred Fish Disk 26

UnHunk	Processes the Amiga "hunk" loadfiles. Collect code, data, and bss hunks together, allows individual specificato of code, data, and bss origins, and generates binary file with format reminiscent of Unix "a.out" format. The output file can be easily processed by a separate program to produce Motorola "S-records" suitable for downloading to PROM programmer. By Eric Black.
C-kermit	Port of the Kermit file transfer program and server.
Ps	Display and set process priorities
Archx	Yet another program for bundling up text files and mailing or posting them as a single file unit.

Fred Fish Disk 27

ABdemos	Amiga Basic demos from Caroly Schleppner.
NewConvertFD	creates .bmaps from fd files.
BitPlanes	finds addresses of and writes to bitplanes of the screen's bitmap.
AboutBmaps	A tutorial on creation and use of bmaps.
LoadILBM	loads and displays IFF ILBM pics.
LoadACBM	loads and displays ACBM pics.
ScreenPrint	creates a demo screen and dumps it to a graphic printer.
Disassem	Simple 68000 disassembler. Reads standard Amiga object files and disassembles the code sections. Data sections are dumped in hex. The actual disassember routines are set up to be callable from a user program so instructions in memory can be disassembled dynamically. By Bill Rogers.
DvorakKeymap	Example of a keymap structure for the Dvorak keyboard layout. Untested but included because assembly examples are few and far between. By Robert Burns of CA
Hypocycloids	Sprograph, from Feb. 84 Byte.
LinesDemo	Example of proportional gadgets to scroll a SuperBitMap.
MemExpansion	Schematics and directions for building your own homebrew 1 Mb memory expansion, by Michael Fellinger.
SafeMalloc	Program to debug 'malloc()' calls
ScienceDemos	Convert Julian to solar and sidereal time, stellar positions and radial velocity epoch calculations and Galilean satellite plotter. By David Eagle.

Fred Fish Disk 28

ABasic games	by David Addison: Backgammon, Cribbage, Milestone, and Othello
Cpp	DECUS 'cpp' C preprocessor, and a modified 'cc' that knows about the 'cpp', for Manx C.
Shar	Unix-compatible shell archiver, for packing files for travel.
SuperBitMap	Example of using a ScrollLayer, syncing SuperBitMaps for printing, and creating dummy RastPorts.

Fred Fish Disk 29

AegisDraw Demo	Demo program without save and no docs.
Animator Demo	Player for the Aegis Animator files
Cc	Unix-like front-end for Manx C
Enough	Tests for existence of system resources, files, and devices
Rubik	Animated Rubik's cube program
StringLib	
Vt100	VT-100 terminal emulator with Kermit and Xmodem protocols

Fred Fish Disk 30

Several shareware programs. The authors request a donation if you find their program useful, so they can write more software.

BBS	an Amiga Basic BBS by Ewan Grantham
FineArt	Amiga art
FontEditor	edit fonts, by Tim Robinson
MenuEditor	Create menus, save them as C source, by David Pehrson
StarTerm3.0	Very nice telecommunications by J. Nangano

(Fred Fish Disk#30 is free if requested when ordered with at least three other disks from the collection.)

Fred Fish Disk 31

Life	Life game, uses blitter to do 19.8 generations a second.
Mandelbrot	Version 3.0 of Robert French's program.
MxExample	Mutual exclusion gadget example.
RamSpeed	Measure relative RAM speed, chip and fast.
Set	Replacement for the Manx "set" command for environment variables,with improvements.
Tree	Draws a recursive tree, green leafy type, not files.
TxEd	Crippled demo version of Microsmith's text editor, TxEd.
VDraw	Full-featured drawing program by Stephen Vermeulen.
Xicon	Invokes CLI scripts from icon
Ticon	Displays text files from an icon.

Fred Fish Disk 32

Address	Extended address book written in AmigaBasic.
Calendar	Calendar/diary program written in AmigaBasic.
DosPlus1	First volume of CLI oriented tools for developers.
DosPlus2	Second volume of CLI oriented tools for developers.

Executables only:

MacView	Views MacPaint pictures in Amiga low or high res, no sample pictures, by Scott Evernden.
Puzzle	Simulation of puzzle with moving square tiles.
ShowHAM	View HAM pictures from CLI.
Solitaire	ABasiC games of Canfield and Klondike, from David Addison.
Spin3	Graphics demo of spinning cubes, double-buffered example.
Sword	Sword of Fallen Angel text adventure game written in Amiga Basic.
Trails	Leaves a trail behind mouse, in Modula-2

Fred Fish Disk 33

3dstars	3d version of the "stars" program below.
Bigmap	Low-level graphics example scrolls bitmap with ScrollVPort.
Dbuf.gels	Double-buffered animation example for BOBs and VSprites.
DiskMapper	Displays sector allocation of floppy disks.
MemView	View memory in real time, move with joystick.
Oing	Bouncing balls demo
Sproing	Oing, with sound effects.
ScreenDump	Dumps highest screen or window to the printer.
Sdb	Simple database program from a DECUS tape.
Stars	Star field demo, like Star Trek.
TermPlus	Terminal program with capture, library, function keys, Xmodem, CIS-B protocols.
Vt100	Version 2.0 of Dave Wecker's VT-100 emulator, with scripts & function

Fred Fish Disk 34

Alint	Support files for Gimpel's 'lint' syntax checker
Blink	PD 'alink' compaible linker,faster,better.
Browser	Updated to FF 18 'browser', in Manx, with scroll bars, bug fixes.
Btree	b-tree data structure examples
Btree2	Another version of 'btree'
Calendar	Appointment calendar with alarm.
Less	File viewer, searching, position by percent, line number.
NewFonts	Set of 28 new Amiga fonts from Bil Fischer
Pr	Background print utility, style options, wildcards.
Requester	Deluxe Paint-type file requester, with sample.

Fred Fish Fred Fish Disk 35

ASendPacket	C example of making asynchronous I/O calls to a DOS handler, written by C-A
ConsoleWindow	C example of getting the Intuition pointer a CON: or RAW: window, for 1.2, by C-A.
DirUtil	Walk the directory tree, do CLI operations from menus
DirUtil2	Another variant of DirutI.
FileRequester	Lattice C file requester module, with demo driver, from Charlie Heath.
MacView	Views MacPaint pictures in Amiga low or high res, with sample pictures, by Scott Evernden.
Plop	Simple IFF reader program
PopCLI	Sidekick-style program invokes a new CLI, with automatic screen blanking.
QuickCopy	Davenport disk copiers duplicate copy-protected disks.
ScrollPf	Dual playfield example, from C-A, shows 400 x 300 x 2 bit plane playfield on a 320 x 200 x 2 plane deep playfield.
SendPacket	General purpose subroutine to send AmigaDos packets.
SpriteMaker	Sprite editor, can save work as C data structure. Shareware by Ray Larson.
Tracker	Converts any disk into files, for electronic transmission. Preserves entire file structure. Shareware by Brad Wilson.
TriClops 3-D	space invasion game, formerly commercial, now public domain. From Geodesic Publications.
Tsize	Print total size of all files in subdirectories.
Unifdef	C preprocessor to remove given #ifdef'd sections of a file, leaving the rest alone. By Dave Yost
Vtest	VT-100 emulation test program. Requires a Unix system.

Fred Fish Disk 36

Acp	Unix-like 'cp' copy program
Clock	Updated version of clock on disk 15.
Csh	Manx 'csh'-like CLI, history, variables, etc.
DietAid	Diet planning aid organizes recipes, calories
Echo	Improved 'echo' command with color, cursor addressing
FixHunk	Fixs programs to let them run in external memory.
Fm	Maps the sectors a file uses on the disk.
KickBench	Docs, program to make a single disk that works like a Kickstart and Workbench.
Lex	Computes Fog, Flesch, and Kincaid readability of text files.
TunnelVision	David Addison ABasic 3D maze perspective game.
Vc	Visicalc-like spreadsheet calculator program.
Vt100	Version 2.2 of Dave Wecker's telecom program
YaBoing	Oing! style game program shows sprite collision detects

Fred Fish Disk 37

This disk is a port of Timothy Budd's Little Smalltalk system, done by Bill Kinnersley at Washington State University.

Fred Fish Disk 38

CSquared	Sep 86 Sci American, Circle Squared algorithm
FixObj	Strips garbage off Xmodem transfered object files
Handler	AmigaDOS handler (device) example from C-A
Hp-10c	Mimics a HP-10C calculator, written in Modula-2
IFFEncode	Saves the screen as an IFF file
IffDump	Dumps info about an IFF file
Jsh	BDS C-like CLI shell
NewStat	STATUS-like program, shows priority, processes
Reversi	Game of Reversi, version 6.1

UUdecode	Translate binary files to text, Unix-like programs
Vdraw	Drawing program, version 1.14
VoiceFiler	DX MIDI synthesizer voice filer program
Window	Example of creating a DOS window on a custom screen

Fred Fish Disk 39

AnsiEcho	'echo', 'touch', 'list', 'cls' written in assembler.
Display	Displays HAM images from a ray-tracing program, with example pictures.
Driver	Example device driver source, acts like RAM: disk
Xlisp	XLisp 1.7, executable only

Fred Fish Disk 40

Ahost	Terminal emulator with Xmodem, Kermit and CIS B protocols, function keys, scripts, RLE graphics and conference mode.
AmigaMonitor	Dynamically displays the machine state, such as open files, active tasks, resources, device states, interrupts, libraries, ports, etc.
Arc	Popular file compression system, the standard for transiting files
AreaCode	Program that decodes area codes into state and locality.
Blink	'slink' replacement linker, version 6.5
Cosmo	An 'asteriods' clone.
Dg210	Data General D-210 Terminal emulator
DirUtil	Windowed DOS interface program, V 1.4
DOSHelper	Windowed AmigaDOS CLI help program
PagePrint	Prints text files with headers, page breaks, line numbers
PopCLI	Starts a new CLI with a single keystroke, from any program, With a screen-saver feature. Version 2, with
SpriteEd	Sprite Editor edits two sprites at a time
X-Spell	Spelling checker allows edits to files

Fred Fish Disk 41

AmigaVenture	Create your own text adventure programs in AmigaBasic.
Csh	Version 2.03 of Dillon's C sh-like shell. Executable only
Doug	Macro based C debugging package.
DualPlayField	example from CBM, update to Intuition manual
GetFile	Heath's file requester, with source
LatXref	Cross reference of Lattice 3.10 header files
Lines	Line drawing demo program
SetFont	Changes font used in a CLI window
Vt100	Version 2.3 of the VT-100 terminal program.

Fred Fish Disk 42

This disk contains an Amiga version of MicroGNUEmacs.

Fred Fish Disk 43

BasicBoing	AmigaBasic program demos page flipping of a 3D cube
Bom	Demo copy of B.E.S.T. Business Management System.
BbsList	A list of Amiga Bulletin Board Systems
Cc	C compiler frontends for Manx and Lattice C
Copper	A hardware copper list disassembler
InstIFF	Converts Instruments demo sounds to IFF sampled sounds
PopColours	Adjust RGB colors of any screen
SpriteClock	Simple clock is displayed on a sprite above all screens
ST Emulator	Non-serious Atari ST emulator
WBrun	Lets Workbench programs be run from the CLI
Wild	Two Unix shell style wild card matching routines

Fred Fish Disk 44

Icons	Miscellaneous icons
NewIFF	New IFF material from CBM for sampled voice and music files
RayTracePics	The famous ray-tracing pictures, from FF#39, now converted to IFF HAM format for "much" faster viewing.
ViewILBM	Displays normal and HAM ILBM files

Fred Fish Disk 45

Clue	Clue board game
Make	Another 'make', with more features
Pictures	Miscellaneous pictures
Update	Updates an older disk with newer files from another disk
WhereIs	Searches a disk for files of given name

Fred Fish Disk 46

Asm	Shareware 68010 macro assembler, ROM Kernal Manual compatible
CheckModem	'execute' file program detects presence of modem
Egad	Gadget editor from the Programmers Network
Jive	Transforms a file from English to Jive.
MyJio	A binary only copy of Ma tt's alternate runtime library. Author: Matt Dillon
ProffMacros	Subset Berkeley 'me' and 'mm' macros for 'proff'
ValSpeak	Transforms a file from English to Valley Speak.

Fred Fish Disk 47

3D-Arm	Simulation of a robotic a rm, very good graphics, teaching tool, including C source.
Juggler	Eric Graham's stunning HAM animation of a robot juggler
VT-100	Version 2.4 of Dave Wecker's terminal emulator, with Xmodem and Kermit file transfer protocols

Fred Fish Disk 48

Bru	Alpha version of a hard disk file archiver
Comm	Version 1.30 of a terminal emulator with phone directories
Csh	Version 2.04 of Matt Dillion's Unix 'csh'-like CLI replacement, including Lattice and Manx C source
Diskperf	Disk benchmark program for Unix and Amiga
Du	Computes disk storage of a file or directory
MemWatch	Program to watch for programs that trash low memory. It attempts to repair the damage, and puts up a requester to inform you of the damage. From the Software Distillery.
Profiler	A realtime execution profiler for Manx C programs. Includes C source.

Fred Fish Disk 49

Cycloids	Update of electronic spirograph from disk 27
DirUtil	Enhanced version of DirUtil from disk 35
MultiDef	Scans a set of object modules and libraries searching for multiply defined symbols
MyUpdate	Disk update utility with options for stripping comments from C header files, and interactive verification of the updating process
Plot	Computes and displays 3 dimensional functions in hires
Polygon	Moire type pattern generator with color cycling
QMouse	Queries whether a mouse button is pressed. This can give a return code that can customize a startup-sequence based on whether a mouse button was pressed.
Touch	Example of setting the datestamp on a file, using a technique from Commodore-Amiga
Trees	More extensive version of the trees program on Disk 31

Fred Fish Disk 50

Asm	Version 1.1 of a shareware 68000 macro assembler, compatible with the Metacomco assembler. This includes an example startup module and more Motorola mneumonics.
BreakOut	A brick breakout game, uses 3-D glasses
DiskZap	Version 1.1 of a program to edit disks and binary files
FirstSilicon	A smart CLI replacement with full editing and recall of previous commands
Missile	A Missile Command-type game, with sound, in assembler
PerfectSound	Sound editor for a low-cost sound digitizer
Sizzlers	Graphics demos
UnixArc	Version of 'arc' for Unix System V machines, in C
Wombat	Version 3.01 of Dave Warker's terminal emulator

Fred Fish Disk 51

Bison	GNU for Unix 'yacc', working update to disk 4 version
Compress	Update to the file compression program on Disk 6
Cos	"Wheel of Fortune"-type game in AmigaBasic
DiffSsed	Unix-like 'diff and 'ssed' for finding the differences between two files, and then recreating the other, given one file, and the list of differences.
Sq, Usq	Portable versions of the CP/M squeeze and unsqueeze

Fred Fish Disk 52

Assign	Replacement for AmigaDOS 'assign' command in C
Fractal	Makes random fractal terrains
Poly, HAMPoly	Workbench-type demos for making polygons in lores and HAM
MxGads	Example of mutual exclusion gadgets with GadgetText
Tek4010	Tektronix 4010 terminal emulator
VDraw	Versions 1.16 and 1.19 of a Deluxe Paint-like drawing program

Fred Fish Disk 53

Animations	Demo animations with player program for Aegis Animator
ARCre	Creates rename scripts for files with long names, so they can be easily 'arc'ed and un'arc'ed.
ARP	Preliminary AmigaDOS replacements for 'break', 'cd', 'chmod', 'echo', 'filenote' and 'makedir'
Compiler	Not fully ported to the Amiga, this is a 68000 C compiler. It wil produce simple assembly language output, but needs a lot of work.
Spreadsheet	Update with source of the 'vc' spreadsheet on disk 36
TarSplit	Port of program to split Unix 'tar' archives
UUencode	Utilities to encode and decode binary files for ASCII transmission, expanding them by 35 percent Paint-like drawing program

Fred Fish Disk 53

Animations	Demo animations with player program for Aegis Animator
ARCre	Creates rename scripts for files with long names, so they can be easily 'arc'ed and un'arc'ed.
ARP	Preliminary AmigaDOS replacements for 'break', 'cd', 'chmod', 'echo', 'filenote' and ' makedir'
Compiler	Not fully ported to the Amiga, this is a 68000 C compiler. It wil produce simple assembly language output, but needs a lot of work.
Spreadsheet	Update with source of the 'vc' spreadsheet on disk 36
TarSplit	Port of program to split Unix 'tar'archives
UUencode	Utilities to encode and decode binary files for ASCII transmission, expanding them by 35 percent

Fred Fish Disk 54

Hanoi	Solves Towers of Hanoi Problem in it's own Workbench window. by Ali Ozer
ISpell	Port of a Unix screen oriented, interactive spelling checker. (Expansion RAM required) by Pace Willisson
Ing	A Screen of lots of bouncing little windows by Leo 'Bols Ewhac' Schwab
Lav	Displays number of tasks in run queue, averaged over last 1, 5, and 15 minute periods. by William Rucklidge
MIDITools	Programs to play/record through the MIDI I/F. by Fred Cassirer
MoreRows	Program to make the Work Bench Screen larger than normal. by Neil Katin and Jim Mackraz
Tilt	Program to make your Amiga look like it didn't pass vibration testing. by Leo 'Bols Ewhac' Schwab

Fred Fish Disk 55

Csh	V2.05 of Matt Dillon's csh like shell (Modified for Manx C). by Matt Dillon, Modified by Steve Drew
NewStartups	New C Startup modules:
AStartup.asm	with 1.2 fixes and better quote handling.
TWStartup.asm	opens a stdio window, using user specs. by Commodore, posted to BIX by Carolyn Schepper
Palette	Change another program's screen colors. by Carolyn Schepper
PipeDevice	Allows the standard output of one process to be fed to the standard input of another. by Matt Dillon
ScreenSave	Save a normal or HAM mode screen as an IFF file. by Carolyn Schepper
ShanghaiDemo	Demo of the Activision game Shanghai.
SoundExample	A double buffered sound example for Manx C. by Jim Goodnow
Vsprites	A working vsprite example. by Eric Cotton
Vt100	V2.6 of Dave's Vt100 terminal emulator with kermit and xmodem. by Dave Wecker

Fred Fish 56

ClipBoard	Clipboard device interface routines, to provide a standard interface. by Andy Finkle
ConPackets	Demos the use of DOS Packets, ConUnit, etc. by Carolyn Schepper
GetDisks	Program to find all available disk device names and return them as an exec list. by Philip Lindsay
GetVolume	Program to get volume name of the volume that a given file resides on. by Chuck McManis
Icon2C	Reads an icon file and writes out a fragment of C code with the icon data structures. by Carolyn Schepper
MergeMem	Program to merge the MemList entries of sequentially configured RAM boards. by Carolyn Schepper
mCAD	An object oriented drawing program, V1.1 by Tim Mooney

Fred Fish 57

CutAndPaste	Implementaions of Unix cut and paste commands. by John Weald
GraphIt	Progran to plot simple functions in 2 or 3 dimensions. by Flynn Fishman
Juggler	V1.2 of robot juggler animation. Uses HAM mode and ray tracing. by Eric Graham
MouseReader	Shareware program to read text files and view IFF files using only the mouse. by William Betz
Ogre	Game of tactical ground combat in the year 2086. by Michael Caplinger; Amiga port by Hobie Orris
Splines	Program to demonstrate curve fitting and rendering techniques. by Helene (Lee) Taran

Fred Fish 58

ASDG-rrd	Extremely useful shareware recoverable ram disk. by Perry Kivolowitz
BigView	Displays any IFF picture, independent of the physical display size, using hardware scroll. by John Hodgson
EGraph	Reads pairs of x and y value from a list of files and draws a formatted graph. by Laurenece Turner
HyperBase	Shareware data management system. V1.5
MemClear	Walks through the free memory lists, zeroing free memory along the way. by John Hodgson
NewZAP	A third-generation multi-purpose file sector editing utility. V3.0 by John Hodgeson
RainBow	A Mauraider-Style rainbow generator. by John Hodgson
SMUSPlayers	Two SMUS plays, to play SMUS IFF music formatted files. by John Hodgson
View	A tiny ILBM viewer by John Hodgson
WBdump	JX-80 optimized workbench printer that does not use DumpRPort. by John Hodgson

Fed Fish Disk 59

Browser	Update to browser program on disks 18 and 34. S-E
Browser2	Another different browser program. E
Clock	Clock program with fonts, colors. E
Dme	Dillon text editor V1.22 for programmers. E-D
DropCloth	Puts a pattern on the Workbench backdrop. E-D
DropShadow	Puts shadows on Workbench windows. E-D
FixWB	Similar to DropCloth, but doesn't work yet S-D
mCAD	Object-oriented drawing program, version 1.2.2. Much improved over disk 56.
Robotroff	Demo of animated pointers on Workbench. S-E-D
Supermort	General compounding/amortization loan calculator. E-D

Fred Fish Disk 60

Various shareware and freeware programs

Blitz	Memory resident file viewer. Very fast. E-D
BlitzFonts	Makes text output faster. E-D
HandShake	Terminal emulator with VT52/VT100/ VT102support. E-D
Med	Mouse-driven text editor version 2.1. E-D
PrtDrvGen	Generates printer drivers, version 1.1. Source available from author. E-D
Show	Sideshow-like IFF viewer, V2.1. E-D
Uedit	Customizable text editor V2.0. E-D
Ueturbo	Example Uedit setup macros. S-E-D

Fred Fish Disk 61

ATPatch	Patches Transformer to work under AmigaDOS 1.2. S-E-D
FillDisk	Writes zeroes to free blocks on a disk for security. S-E-D
LPatch	Patch for programs that abort when loading under AmigaDOS 1.2. S-E-D
MicroEmacs	Conroy MicroEmacs V3.8b, newer than disk 22. S-E-D
PearlFont	Like Topaz, but rounded edges.
Terrain	Generates fractal scenery. S-E-D
VSprites	Makes 28 Vsprites, from P&E Book.

Fred Fish Disk 62

This is a port of the Unix game 'Hack', by the Software Distillery, version 1.0.3D.

Fred Fish Disk 63

This is a port of the Unix game 'Larn', by the Software Distillery, version 12.0B.

Fred Fish Disk 64

This is an offical IFF specification disk from Commodore, an update to disk 16.

Fred Fish Disk 65

Bawk	Unix text processor, like 'awk'. Doesn't work, but source is included. S-E-D.
MWB	Example of rerouting Workbench window open calls to another custom screen. Version 1.01, S-E-D
CloseWB	Example for closing a custom Workbench screen. S-E-D
Cookie	Generates one-line fortune-cookie aphorisms. S-E-D
JTime	Build-your-own mouse port clock.
MenuBuilder	Creates C source files for menus, based on text descriptions. S-E-D.
NewPackets	CBM tutorial on new packets and structures in AmigaDos 1.2.
PascalToC	Pascal to C translator, not so great. S-E-D
Prep	'ratfor'-like FORTRAN preprocessor. S-E-D
RunBack	Starts programs from CLI, allowing CLI window to close. E-D
SunMouse	This program automatically clicks in windows when the mouse is moved over them. Version 1.0, E-D

Fred Fish Disk 66

AmScsi	Preliminary plans for a SCSI disk controller board.
Asm68k	Macro assembler, version 1.0.1. E-D
Assigned	Example for avoiding DOS insert-disk requester, by scanning the list of 'assign'ed names. S-E-D
Dk	Pretends to eat away at CLI window. S-E-D
Flip	Flips whole screen as a joke. S-E-D
Foogol	Foogol cross-compiler generates VAX assembly code. S-E-D
Free	Prints amount of free space on all drives. S-E-D
MallocTest	malloc/free memory test program. S-E-D
Melt	Pretends to melt the screen. S-E-D
Nart	Graphic flying string demo. S-E-D
Purty	Easy way to set printer attributes from Workbench. E-D
RayTracer	Simple ray tracing program. E-D
SendPackets	Updated CBM examples of packet routines on disk 35. S-E-D
SnapShot	Memory resident screen dump. E-D
TagBBS	Shareware BBS system, version 1.02.

Fred Fish Disk 67

AmCat	Shareware disk cataloging program.
AmigaSpell	Shareware Intuition spelling checker, V2.0. E-D
Bouncer	3-D bouncing ball written in MultiForth, S-E-D
Comm	Terminal program version 1.33, E
Dux5	Another version of DirUtil. S-E-D
HexCalc	Hex, octal, & decimal calculator. E-D
Icons	Various big and alternate image icons.
Mandala	Mandala graphics and sound. E
PersMail	Demo shareware personal file manager.
RSLClock	Menu bar clock version 1.3. E-D
RTCubes	Graphics demo of 3D cubes. E-D
Wheel	"Wheel of Fortune"-type game, in AmigaBasic

Fred Fish Disk 68

This is version MG 1b of the MicroGNUEmacs. Source and executable areincluded, as well as source for other computers besides the Amiga.

Fred Fish 69

Asm68k	Macro assembler, v1.0.3, E-D
BitLab	Bitter exploring program, in C, S-E-D
Conman	Replacement console device handler adds editing and history to any application that uses CON:, v0.9, E-D
Console	Replacement console routines, in C, S-E-D
Dk	Decays the screen bit by bit, update to disk 66, in Modula-2, S-E-D
Frags	Displays memory fragmentation listing the size of free memory blocks, in C, S-E-D
IconType	Change the type of an icon, in C, S-E-D
Make	'make' in Manx C, S-E-D
MonProc	Monitors processes for packet activity, in C,S-ED
MouseClock	Turns mouse pointer into a digital clock,in C,SED
Sb	Browses system structures, from Transactor magazine, v1.0, in C, S-E-D

For PDS orders, Please use form on page 106

Spew	Generates 'National Enquirer'-type headlines from rules file. In C,S-E-D
Spool	Three programs to demonstrate multitasking and spooling in a printer spooler. In C, v1.2, S-E-D
Wc	Counts words ala Unix 'wc', but faster, in C,S-ED

Fred Fish 70

This is a disk of shareware programs.

AmigaMonitor	Explores state of the system, v1.13
Arc	Standard file compressor and librarian, v0.23, a port of MS-DOS v5.0. E-D
BlackBook	Phone book program.
DoTil	Intuition-driven file manipulator program, v2.0.
GravityWars	Game of planets, ships and black holes, v1.03.
Jobs	Alternate user interface to CLI and Workbench, v2.1.
Lens	Magnifies area around mouse, shows it in a window, v1.0.
Life-3d	3D version of the classic cellular-automaton game, v1.2.
Logo	Logo language interpreter
SetKey	Demo keymap editor, v1.0
Vpg	Makes displays for aligning video monitors, v1.0.

Fred Fish 71

AirFoil	Makes airfoils using the Joukowski transformation, in C, S-E-D
Amiga Basic	Miscellaneous programs including 3D plot program, a kaleidoscope, C-A logo drawing program file comparison utility string search program, S-E-D
Blocks	A variation of "lines", but with variable color blocks. E-D
Comm	Great terminal program, v1.34, E-D
DiskX	Utility for exploring file system.E-D
Fpic	Simple image processing program that operates on IFF pictures, with several filters, merging images, E-D
IconMk	Makes icons for files , v1.2a, E-D
Icons	New icons
NewFonts	Two new fonts; 'shalt18', an electronic ciruit element font, and 'ibm5', a PC-like font.
PetCLI	An AmigaBASIC CLI shell program.
PWDemo	Demo of the commercial product PowerWindows,v1.2. It aids creation of custom windows, menus, and gadgets, giving C or assembly source. E-D
Rot	Creates and animates 3-D objects, v0.5, E-D
TimeSet	Sets time from Workbench, E-D

Fred Fish 72

This is a disk of IFF pictures.

Fred Fish 73

Add	Customizes existing program menus with Amiga-key shortcuts. Also includes 'until', which waits until a given window is created. Shareware, in C, S-E-D.
AutoIconOpen	Fools WB into thinking mouse has double-clicked icons. In C, S-E-D
Dio	Generic Exec device interface code for opening libraries, getting multiple I/O channels, asynchronous operations, etc. In C, S-E-D.
Dissolve	Slowly displays IFF files, ala Nov 86 Dr. Dobb's program. In C, S-E-D
DTerm	Flexible, reprogrammable terminal program v1.10, E-D
Expose	Re-arranges windows so that at least one pixel of menu bar gadgets are exposed. In C, S-E-D.
Lit	Scans a text file, converts to C-style printable strings.C,v2.0, S-E-D
Lmv	"Long Movie", program views series of IFF picts in quick succession, upto 19 fps. Shareware, E-D
MouseOff	Mouse pointer disappears after ten seconds of non-use. In C, S-E-D
ParOut	Examples of controlling parallel port with resources instead of the PAR: device.In C,S-E-D
PenPalFont	Child-like font.
RunBackGround	Similar to RunBack on disk 66, runs program from the CLI allowing the CLI window to close. In C,S-E-D
SnapShot	Screendump utility,update FF 66.E-D
TypeAndTell	Example installs a device handler before Intuition, and speaks each key as it is pressed. In C and assembler, S-E-D
Xplor	Prints info about system lists, in assembler,S-E-D

Fred Fish Disk 74

Cled	Edits and recalls CLI commands, v1.3, E-D
Control	Intercepts graphic printer dump calls and accesses color map, width,and screen resolution. C,S-E-D
Dme	Simple WYSIWYG text editor for programmers,v1.25. Update of FF 59.E-D
DropShadow	Workbench dropshadows, v2.0. Update to disk 59. E-D
Funds	AmigaBASIC program tracks mutual or stock p-D
Less	Text viewing program, like Unix "more", v1.1, update to disk 34. S-E-D
Makemake	Scans C source files and constructs a vanilla 'makefile' in the current directory. S-E-D
mCAD	Object-oriented drawing prog. v1.2.4, update to FF 59.Shareware, E-D
Random	Simple random number generator in C. S-E-D
TDebug	Monitors devices by intercepting Exec SendIO() and DoIO() vectors, in C, v1.0, S-E-D
Units	Converts measurements in different units, includes "chart" option, in C, S-E-D
XCopy	Replacement for AmigaDOS 'copy', doesn't change the date, uses Unix wildcards. E-D

Fred Fish Disk 75

Bezier	Play with Bezier curves points and granularity, S-E-D
BSplines	Play with b-splines, as above, S-E-D
Comm	C source for Comm terminal program v1.34. S-E-D
Copy	Replacement 'copy' command v1.0, preserves date, in C, S-E-D
Diff	Simple 'diff' in C, S-E-D
DuM2	Another DrUtil in Module-2, v1.5, S-E-D
Eless	Fast 'dir' program in C, S-E-D
Fd	Faster 'eless' in C, S-E-D
HardCopy	Sends a transcript of a CLI session to a file, in C, S-E-D
MouseOff	Update to disk 73, turns off mouse pointer, S-E-D
SetFont	Changes the font in a Workbench screen, v2.0, S-E-D
SpeedDir	Another fast 'dir', in assembler, S-E-D

Fred Fish Disk 76 & 77

These are disks 1 and 2 of Chris Gray's Draco distribution for the Amiga. Draco is a compiled, structured language reminiscent of both C and Pascal. A full interface to AmigaDOS and Intuition is supplied. Be sure to get both disk 76 and 77.

Fred Fish Disk 78

Cycles	Cycle game like "Tron", v1.0, E-D
EOMS	Experts Only Mercenary Simulator game, E-D
MandelVroom	Mandelbrot generator wth enhanced palette controls, fixed/floating point, presets, v1.50, in Manx C, S-E-D

Fred Fish Disk 79

AsmTools	CLI tools in assembler: echo, loadit, mounted, setace, why; S-E-D
AssignDev	Give devices multiple names, in C, S-E-D
AuxHandler	Example of a dos handler that allows use of a CLI via the serial port. Includes source. Author: Steve Drew
Cmd	Redirects printer output to a file, in C, S-E-D
Info	AmigaDOS 'info' replacement, in C and assembler, S-E-D
Kill	Removes a task and its resources, in C, S-E-D
M2Error	Displays errors from TDI Modula-2 compiles, S-E-D
MonProc	Update to process packet program from disk 69, in C, S-E-D
Mounted	Program for testing if a drive is present, in a script In C, S-E-D
Nro	Another 'roff'-style text formatter, in C, S-E-D
ParTask	Finds parent task, in C, S-E-D
QueryAny	For scripts, asks a question, accepts Y/N, gives return code. In assembler, S-E-D
ScnSizer	Resets preferences settings for screen size, in C, S-E-D
SharedLib	Example of a shared library, in C and assembler, S-E-D
Task	Simple CreateTask() example in C, S-E-D
Uw	Unix Windows client v1.0, in C, S-E-D
Who	Lists tasks on ready and wait queues, in C, S-E-D

Fred Fish Disk 80 (see Fred Fish 90)

Fred Fish 80 has been withdrawn due to copyright problems.

Fred Fish Disk 81

Asm68k	V1.1.0 of a macro assembler
AutoFacc	Shrinks the FACC window and moves it to the back
Brushes	53 custom IFF brushes of electronic symbols
CheckIFF	Checks structure of an IFF file CledV1.4 update to disk 74 of a simple command line editor
Conman	Replaces console handler to add editing and history to many programs
Fonts	Miscellaneous fonts
Icon	V6.0 of the Icon programming language
KeyLock	Freezes the keyboard and mouse until pass word entered.
ScatDisplay	hack created from "fng"
Smush	Smushes an IFF file.
Target	Each mouse clikc becomes a gunshot

Fred Fish Disk 82

AdventureA port of the classic Crowther and Woods Adventure game	
AmicTerm	V0.50 of a telecommunications program, with scripts, redial, beeps, enhanced file requester
D2D-Demo	Demo version of Disk-2-Disk from Central Coast Software
DX-Synth	Voice filer program for Yamaha DX series synthesizers, update to disk 38
DiskMan	V1.0 of another DrUtil program
Icons	Miscellaneous new icons
Pan	Universal MIDI patch panel, v1.2
Rocket	Another Workbench hack, plays Lunar Lander
Sand	Game of sands following your pointer.

Fred Fish Disk 83

This disk contains a demo version of TeX from N Squared. It is limited to small files, and the previewer can only display ten pages or less, and only a small number of fonts are provided.

Fred Fish Disk 84

AudioToolsPrograms from Rob Peck's July/August Amiga World article	
BirLab	Bitter experimentation program, V1.2, update to disk 69
Ed	Simple editor, similar to Unix 'ed', based on the editor in Software Tools.
GravityWars	Game of planets, ships and black holes, v1.04, update to disk 70.
HunkPad	Adds legal padding to executables for Xmodem transmission.
PipeHandler	An AmigaDOS pipe device which supports named pipes and taps. V1.2
PopCLI	V3.0 of a hot-key to invoke a CLI window, with screen blanker, update to disk 40.
Requester	Update to disk 34 of a file requester similar to DPaint.
ScottDevice	V33.1 of a 'mount'able MicroForge SCSI driver.
Viacom	Another Schweb hack, makes TV-like static on screen.

Fred Fish Disk 85

Csh	V2.06 of Dillon's 'csh'-like shell
FileReq	Source to wildcard file requester
Hide	Hides expansion memory from programs
ImageTools	Shareware tools to manipulation IFF images
LowMem	ServerShared library to aid in low memory situations
Plot6	A star plotting program with source.
RawIO	Example of setting raw mode on standard input
Rocket	Lunary Lander for Workbench, with source.
VMore	"more"-like text viewing utility, v1.0 with source.
Vnews	Simple Unix news reader.

Fred Fish Disk 86

AutoPointAuto-selects window under the mouse pointer, with screensaver.	
ClickToFront	Double-clicks in window brings it to front, v1.1, S-E-D
Cmd	V3.0 of a tool to redirect printer output to a file.
FileIISG-Demo	Demo of Softwood File IIsg, a database manager with sound and graphics.

Fred Fish Disk 87

AdvSys	Adventure system from Byte May 1987, v1.2 E-D
AutoIconOpen	Fools Workbench to open disk icons, V1.2 update to disk 73, S-E-D
Ciaz	Converts IFF files to PostScript, V2.0, S-E-D
Commodi	tesMackraz's Commodities Exchange, an exec library to manage the input handler, v0.4
Diff	Update to disk 75 of Unix-like 'diff', S-E-D
Dme	V1.27 of Dillon's text editor, update to disk 74 , E-D
DropShadow	V2.0 of program that puts shadows on Workbench, S-E-D
Elb	Shared library example in Manx C.
ID-Handler	An AmigaDOS device handler that generates unique identifiers, V1.0, S-E-D
Install	Alternate AmigaDOS 'install' programs, S-E-D
MemWatch	Waits for low memory trashing, V2.0, S-E-D
MovePointer	Moves pointer to given location, S-E-D
MoveWindow	Move window to given locaton, S-E-D
MunchingSq	Munching Squares hack, S-E-D
PalTest	Example shows test to see if this is a PAL machine, S-E-D
Sc	Generates random scenery, S-E-D
Tek4695	Tek4695 printer driver
WBDualPF	Example of dual-playfield screen, update to disk 41, S-E-D
WarpText	Fast text rendering routines, S-E-D
Yaff	rExample IFF reader, S-E-D
Zoo	A file archiver like "arc", v1.42A, E-D

Fred Fish Disk 88 (see Fred Fish 89)

Fred Fish Disk 88 has been removed due to copyright problems

Fred Fish Disk 89 (replaces Fred Fish 80)

DirMaster	Disk catalogue program, V1.0a, E-D
FuncKey	Shareware function key editor, V1.01, E-D
MFF-Demo	Demo of MicroFiche Filer database program
ScreenShift	Adjust screen position like Preferences, S-E-D
Snake	Bouncing squiggly lines demo, S-E-D
AutoEnguirer	screen contraption requester improvement S-E-D
DemoLition	Display Hack S-E-D

Fred Fish Disk 90 (replaces Fred Fish 80)

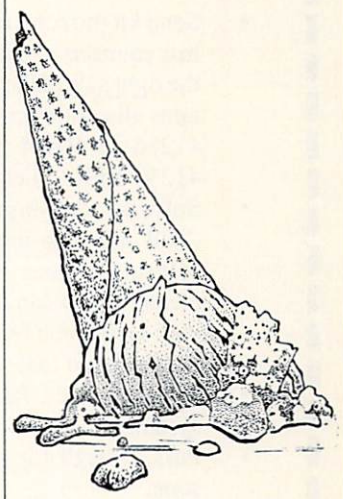
AmiGazer	Night sky viewer of 1573 stars, set date, time, day. E-D
CardFile	AmigaBasic card file study aid. E-D
Conman	Console handler replacement gives line editing and history to most progs, v0.98,E-D
MandelVroom	Slight update to disk 78 Mandelbrot program, E-D
NewDemos	Replacements for lines and boxes demos that take less CPU time, E-D
Othello	Game of Othello, E-D
PrinText	Displays text files with gadgets, speech, IFF display, v1.2, E-D
PrtDrvGen	Automatic printer driver generator, v2.2b, E-D
RainBench	Cycles colors of Workbench backdrop or text. E-D
ShortCut	Makes single-key shortcuts for entering commonly typed CLI commands, as well as custom macros. E-D
ShowPrint	Displays and prints IFF pictures of all sizes, and controls printer output styles, v2.0 E-D
Sizzlers	Graphics demos, v1.7.0, E-D
Timer	Small Workbench timer counts time and $/ minute, E-D
Tools	Inovstronics tools: a memory editor, memory disassembler, ASCII chart, and csiculator. E

To Be Continued.....

In Conclusion

To the best of our knowledge, the materials in this library are freely distributable. This means they were either publicly posted and placed in the Public Domain by their Author, or they have restrictions published in their files to which we have adhered. If you become aware of any violation of the author's wishes, please contact us by mail.

•AC•



Oops . . .

Corrections!

Last month's review of **Microbotics' Starboard-2** (AC V 2.9, p. 40) stated that the Starboard-2 "does not adhere to the Zorro standard."

Starboard-2 does, in fact (and always has), use the Zorro standard.

We apologize for any inconvenience caused by our error. •AC•

For PDS orders, Please use form on page 106

Index of Advertisers

LAND OF LEGENDS™

The first in our quest master series! Land of Legends' mystery and excitement awaits your every move as you sojourn through one adventure after another. Truly the most thrilling of all Dungeon & Dragon type adventures.

PLANETARIUM

For The Serious Student of Astronomy

For the serious student of astronomy, PLANETARIUM's features include over 9,000 stars down to the 7th magnitude, all hemispheric and airborne views, latest NASA stellar and planetary ephemerides, Skies from 9999 B.C. to 9999 A.D., programmable to incorporate new discoveries, accurate celestial representations, and optionally displayed names and patterns of constellations. PLANETARIUM, an astronomical delight, get yours today!

BlackJack Academy

MICRO-VICE SERIES

Everything you ever wanted to know about the game of BlackJack. For the novice learning the game, or the pro polishing skills, BlackJack Academy offers both high powered instruction and realistic game play. Develop your skills, and have fun playing BlackJack with BlackJack Academy. Now available at your local software dealer!

OTHER PRODUCTS FROM MICROILLUSIONS

- **THE FAERY TALE ADVENTURE™**
  The HOTTEST game for the Amiga!, soon to be released in C64/128, a must game for every user!
- **ROMANTIC ENCOUNTERS AT THE DOME™**
  A true to life adult experience for men or women.

NEW

- **DISCOVERY™ DATA DISKS**
  Science/Math/Geography/Spelling/ Language/History/Trivia/Social Studies, Now Available!
- **ONE TO ONE SERIES™**
  ARCADE GAMES: Galactic Invasions,™ Fire Power,™ and Turbo™ Arcade titles now available!

These products are now available, or are being developed for the Amiga, and will soon be available in other formats (C-64/128, IBM/PC, Apple).

microillusions™

17408 Chatsworth St., Granada Hills, CA 91344, inside CA 818/360-3715 • outside CA 800/522-2041 • FAX 818/360-1464

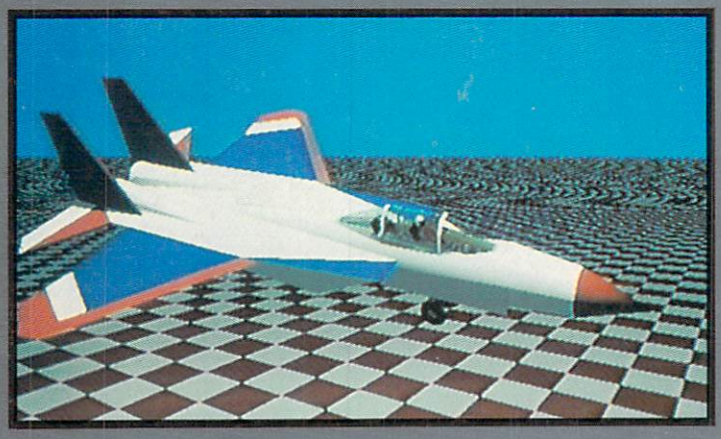
ROME WASN'T BUILT IN A DAY, UNTIL NOW . . .

Create your own universe with SCULPT 3-D™

SCULPT 3-D brings the power of 3 dimensional solid modeling and ray tracing to the Amiga. Imagine an image: choose a color, a shape, a texture. Spin it, rotate it, extrude it into the third dimension. Pick a camera lens, set your lights, and let SCULPT 3-D create a three dimensional picture complete with shadows, reflections, and smooth shading. All in 4096 colors with true edge to edge overscan video. Easily! Automatically! Change your mind? Change the colors, textures, camera or lights in seconds and create a new image. The only limits are the boundaries of your imagination.

"I haven't had this much fun with a program since Deluxe Paint II." John Foust of Amazing Computing.

"Performance previously only available on mini and mainframe computers." Info Magazine.



Now animate your universe with ANIMATE 3-D™

Enter the fourth dimension, time. Choreograph the free flowing and simultaneous movement of objects, lights and camera through space and time. Details of object rotation, camera movements, timing and action are controlled in an easy to use graphical interface or through a simple script language. Individual objects can be linked to orchestrate complex hierarchial movements that simulate live action. Quick check wireframe playback preview's your final production: storable as a compressed animation file playable from RAM, or recorded on videotape. Additional output options include single frame VCR control or image rendering to a frame buffer card. Animations can incorporate either solid modeling or ray tracing. ANIMATE 3-D is quite simply the most powerful and easy to use animation program available for the Amiga.

Expand your universe with the BYTE BOX™

Your Amiga 500 deserves the best you can give it. More memory for more powerful applications, faster performance, better graphics, and RAM disk storage. It deserves a memory expansion system that lets you add additional memory as you need it. In easy to install and easy to afford increments. The included memory verify software provides a visual check whenever you add additional RAM. The BYTE BOX is available in a variety of configurations from 0MBytes to 2MBytes of RAM.

- Easy to install
- Fully Auto-Configure
- Fast memory that's truly fast
- Has its own power supply
- Fully tested and ready to use
- Zero wait state design
- Low profile case
- Memory check software

BYTE by BYTE™ CORPORATION

Aboretum Plaza II 9442 Capital of Texas Highway North Suite 150 Austin, TX 78759 (512) 343-4357